August 2021

A System for Programming Anisotropic Physical Behaviour in Cloth Fabric

Abhinit Sati
Clemson University, asati@clemson.edu

A System for Programming Anisotropic Physical Behavior in Cloth Fabric

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
Abhinit Sati
August 2021

Accepted by:
Dr. Victor Zordan, Committee Chair
Dr. Ioannis Karamouzas
Dr. Yin Yang

Abstract

We propose a method to alter the tensile properties of cloth in a user defined and purposeful manner with the help of computer controlled embroidery. Our system is capable of infusing non-uniform stiffening in local regions of the cloth. This has numerous applications in the manufacturing of high performance smart textiles for the medical industry, sports goods, comfort-wear, etc where pressure needs to be redistributed and the cloth needs to deform correctly under a given load. We make three contributions to accomplish this: a decomposition scheme that expresses user-desired stiffness as a density map and a directional map, a novel stitch planning algorithm that produces a series of stitches adhering to the input stiffness maps and an inverse design based optimization driven by a cloth simulator that automatically computes stiffness maps based on user specified performance criteria. We perform multiple tests on physically manufactured cloth samples to show how embroidery affects the resultant fabric to demonstrate the efficacy of our approach.

Acknowledgments

I would like to sincerely thank Dr. Victor Zordan and Dr. Ioannis Karamouzas for their constant support, for giving me this wonderful opportunity and introducing me to the exciting field of computational fabrication. Their guidance and encouragement is what has made this work possible. I would also like to thank them for the countless hours they spent in helping me understand difficult concepts and all the amazing innovative discussions we have had over the last 1.5 years.

I also thank my parents for supporting me through thick and thin, and encouraging me to pursue studies away from home.

Table of Contents

List of Figures

Chapter 1

Introduction

Advancements in textile manufacturing technology have reduced a lot of manual labour that went into the textile production process and promise a future where entire garments can be produced by a single machine. These whole garment knitting machines are computers that read in knit instructions to fabricate garments and look identical to how a designer chooses to visualize them on their desktop computer or laptop. They also reduce the need for complex sewing of flat fabric panels to produce the final 3D garment and a host of other post processing steps which require skilled human intervention. Soon, all functional textiles consumed globally might be manufactured with this process. Despite all these major advances, virtually all manufactured fabric products exhibit the same mechanical properties throughout their surface. This is a byproduct of the manufacturing process itself, since all of them (to the best of our knowledge) are woven/knitted with the same yarn material along both the warp and weft directions. Variations in stretch can be introduced but they remain constant along the respective axis, e.g a 4 way stretch fabric can be more flexible along the horizontal direction and less along the vertical direction or vice-versa. In this work, we introduce a series of techniques for altering tensile properties of fabric and provide a solution that gives flexibility over how those properties can be controlled precisely over local regions of the fabric surface. The result is a piece of fabric that stretches arbitrarily in arbitrary directions across it's surface (Figure 1.1). Our aim with this work is to build next generation "smart-stretch" fabric that can be programmed to adhere to desired physical properties specified by a user.

Our work is based of off [24] and colleagues approach to solving a similar set of problems, whose work was only concerned with modifying isotropic properties of the fabric surface. The

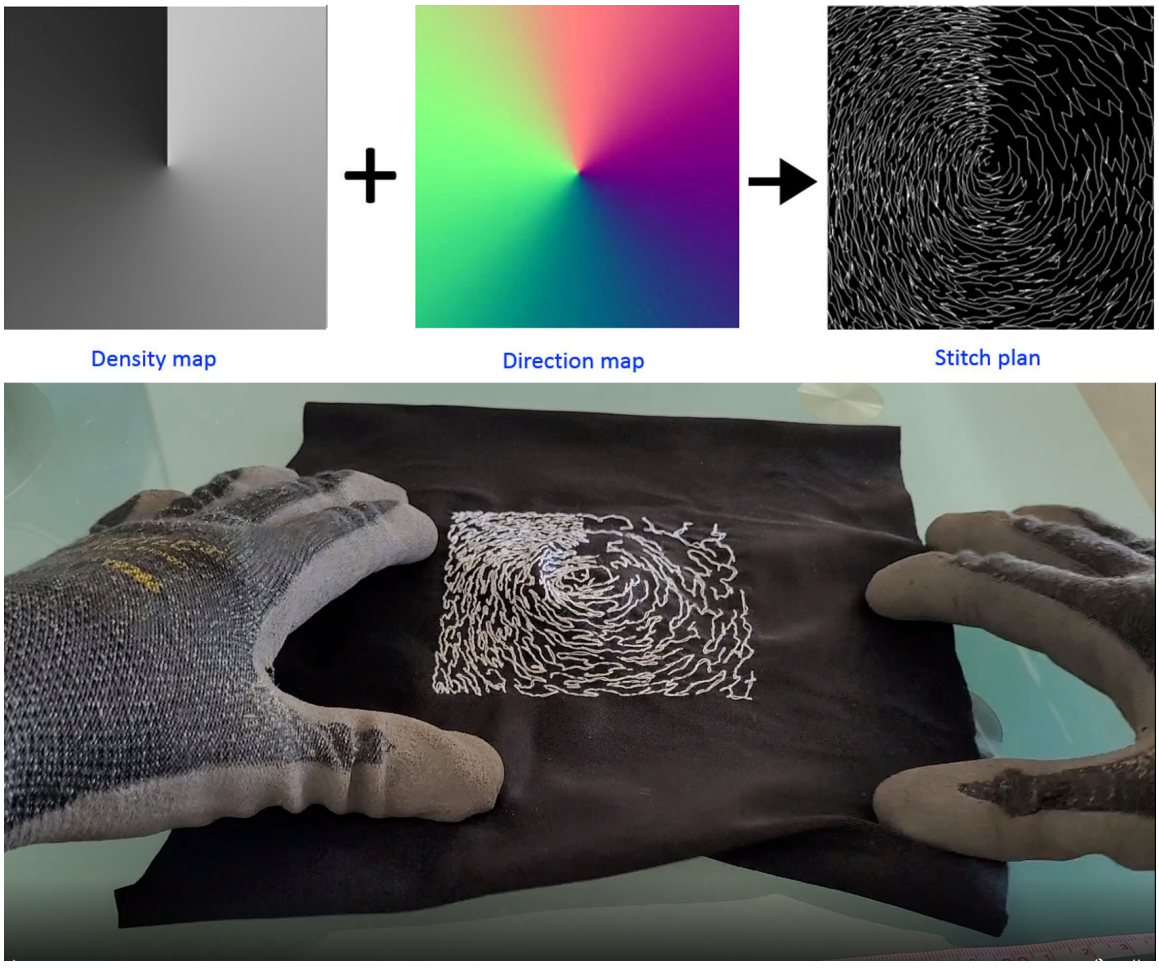

Figure 1.1: Direction and density combine to create final embroidered cloth with rotational stretch patterns and unique tensile properties across the resulting fabric surface.

solution we have developed aims to extend their work to also incorporate anisotropic properties into the cloth. [24] model the problem of producing isotropic properties as a graph problem where every vertex sampled on the fabric needs to be visited exactly once, akin to solving the Travelling Salesmen Problem in conventional graph theory. An embroidery machine is employed to put down stitches connecting these vertices on the fabric surface, altering its properties in the process, where every stitch made impedes stretching when pulled. A series of stitches being embroidered on the fabric surface can thus produce fabric samples that can stretch with variable stiffness in different regions. The stretch along the shear direction can also be altered. [24] use a density map to control stiffness of local regions. A density map is a grayscale image specifying how *stiff* a region needs to be. Darker regions imply more stiffness (Figure 1.1). When discretizing the fabric to sample vertices, more vertices are added to the regions containing a darker shade. Because all vertices need to be touched at least once in order to cover the whole surface, more stitches are put down in darker regions, which makes the area occupying those regions more stiff than others.

The path planner designed by [24] is optimized for producing stitches that make 90 degree angles with each other. This helps distribute stitches uniformly in all directions and prevents any kind of bias towards a specific direction. The result is a stitch plan that produces uniform stiffening in all directions on the fabric surface. It is important to note that the stiffening can change from region to region, but the stiffness at a certain point is the same when pulled in any direction. Hence, this strategy produces isotropic cloth samples. To move to a solution that produces anisotropic samples, it is imperative to control stitch direction with maximum precision. This means certain stitches need to be biased in a specific direction, thus promoting more stiffness along those directions leading to the formation of anisotropic samples. We encode preferred stitch direction information in an RGB image called a *normal map* or *direction map* and interpret it as a vector field of directions (Figure 1.1). The direction map shown in Figure 1.1 encodes a vector field that looks like a series of concentric circles, giving the corresponding stitch plan a similar look. This extra piece of information about preferred directions and the density map from before are the inputs our path planner is given. There are quite a few differences between our solution and the one proposed by [24] and these will be highlighted in greater detail in Section 3. Our solution also has the ability to smoothly move between complete anisotropy to complete isotropy.

We also introduce an inverse design based map optimization into the pipeline. Generating density and direction maps by hand isn't very intuitive for most practical use cases, because it is

difficult to predict the behaviour of the cloth given a stitch pattern. We will showcase two simple examples to demonstrate the process of automating map making. The approach is driven by the following philosophy - the user specifies how the cloth should behave based on given external forces acting on it, the algorithm then computes the best density and direction maps needed to produce the desired behaviour. We describe how a density and direction map affect the simulated cloth in Section 3. Possible use cases of this technique include designing an ergonomic chair where the cushion deforms such that the person sitting on it has their back upright. Designing custom shoe insoles that respond appropriately based on a persons gait or how they run is another possible use case (Figure 1.2).

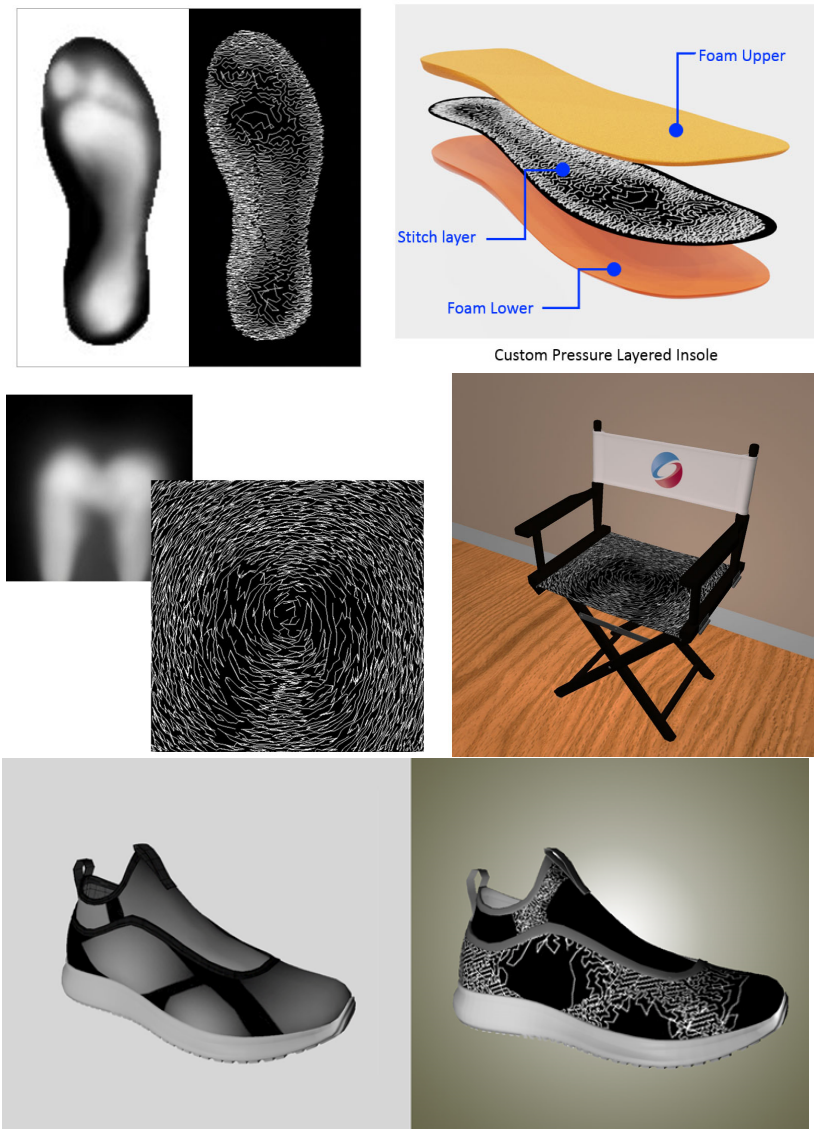

Figure 1.2: Potential real-world examples of the proposed technology. (Top) Concept example for customized insole from foot impression. (Middle) Example of director's chair that supports pressure profile shown. (Bottom) Shoe example where stitches create a comfortable slick and low cost upper that is also supportive like a lace or buckle.

Chapter 2

Related Work

There has been a considerable amount of progress in computational techniques employed to alter physical properties of objects made of various materials. The work of [27] demonstrates how to obtain user-desired Poisson ratio and Young's modulus of 3D printed micro-structures, allowing the creation of both isotropic and an-isotropic structures. [29] and [30] demonstrate something similar, but with rod networks forming aesthetically pleasing patterns. The work of [35] can alter mechanical properties at the macroscopic level by doping silicone with liquid using a filament-based 3D printer. We also see how [20] use procedural Voronoi foams to control the isotropic rigidity/flexibility of their fabricated models. They extend this idea in [21] to fabricate orthotropic models by employing graph algorithms to optimally orient procedural micro-structures. In the domain of textiles, other researchers have solved a series of similar problems, including smart embroidery [31], textiles covering 3D shapes [19], 3D printing using felted fabric [28], user-assisted 3D knitting [15, 16], and 3D weaving [34]. McCann and colleagues [22, 25, 26, 17] have formalized machine knitting by developing knit compilers and several other techniques that convert 3D meshes to knit patterns that can be executed by industrial knitting machines. The visual knitting programming interface proposed by them lets non experts design custom patterns on the surface of knitted fabric produced by the machine. However, none of these techniques have the ability (or intend to) to control stiffness of the resulting fabric to the extent that our method can. Closest to our work is [10], whose aim was to create 3D shapes out of flat pretensioned material embedded with frustum shaped tiles which snap and meet at the right positions on actuation. The base pretensioned material wasn't modified purposefully in their work (is uniformly stretched), and the focus instead was to optimize the locations, shape and

orientations of the tiles for the best possible match of the input target 3D shape upon actuation.

In the past, [24] have been successful at producing patterns that control fabric stiffness uniformly at the stitch level by solving a path planning problem modelled as a Travelling Salesman Problem (TSP). Our work on the other hand, tries to tackle the non-uniform or anisotropic stitch planning problem that was unexplored in the previous work. A piece of fabric subject to certain forces and stretching can thus acquire a desired deformation more efficiently, similar to the work of [36] where flat fabric panels are generated from a 3D shape and sewn together for casting.

[24] and colleagues, inspired by [27] and [30] developed a solution using similar metamaterial altering techniques. The idea was to stack a series of embdroidered blocks (EB) on the fabric surface with different densities. For low density regions, the EB would have fewer horizontal and vertical accordian patterns to reduce stiffness. The number of horizontal and vertical stitches is set to be equal to produce isotropic stiffening. An immediate drawback of this approach is the inflexibility of the EB. Since the EB size has to be fixed before the embroidery process begins (10X10 was the chosen size for the example) the discretization becomes extremely coarse and occupies a fixed area on the fabric surface. To remedy these issues, [24] came up with the alternate stitch planning approach. The stitch planning approach to embroidery provides more granular control at finer resolutions which produces more robust designs.

Chapter 3

Algorithm Design and Methods

We divide the problem statement into two different parts (refer to Fig 3.1 for an overview of the approach), each of which are described separately in the following sections. The first section covers the stitch planning aspect of the system followed by a detailed description of the inverse design based optimization.

3.1 Stitch Planning

We model the stitch planning problem as a variant of the Travelling Salesman graph problem, where each sampled point on the fabric needs to be visited at least once in order to create a network of stitches. Note that this modelling differs from the traditional definition of the problem in graph theory, as we want to visit every point at least once and not exactly once (akin to the Minimum Spanning Tree problem). This gives us flexibility to generate a network (a graph or a tree) as input to the embroidery machine in contrast to a path where every point can be touched exactly once. We will see in the following sections how this benefits the stitch planning algorithm and helps solve some of the issues faced by the stitch planner in the previous work [24].

3.1.1 Sampling - Discretizing the fabric

This step of the process generates the vertices of the graph or stitch plan we produce as output. These points are strategically placed according to a density map (Fig 3.2).

The density map is a grayscale image, where darker (channel values that are smaller) regions

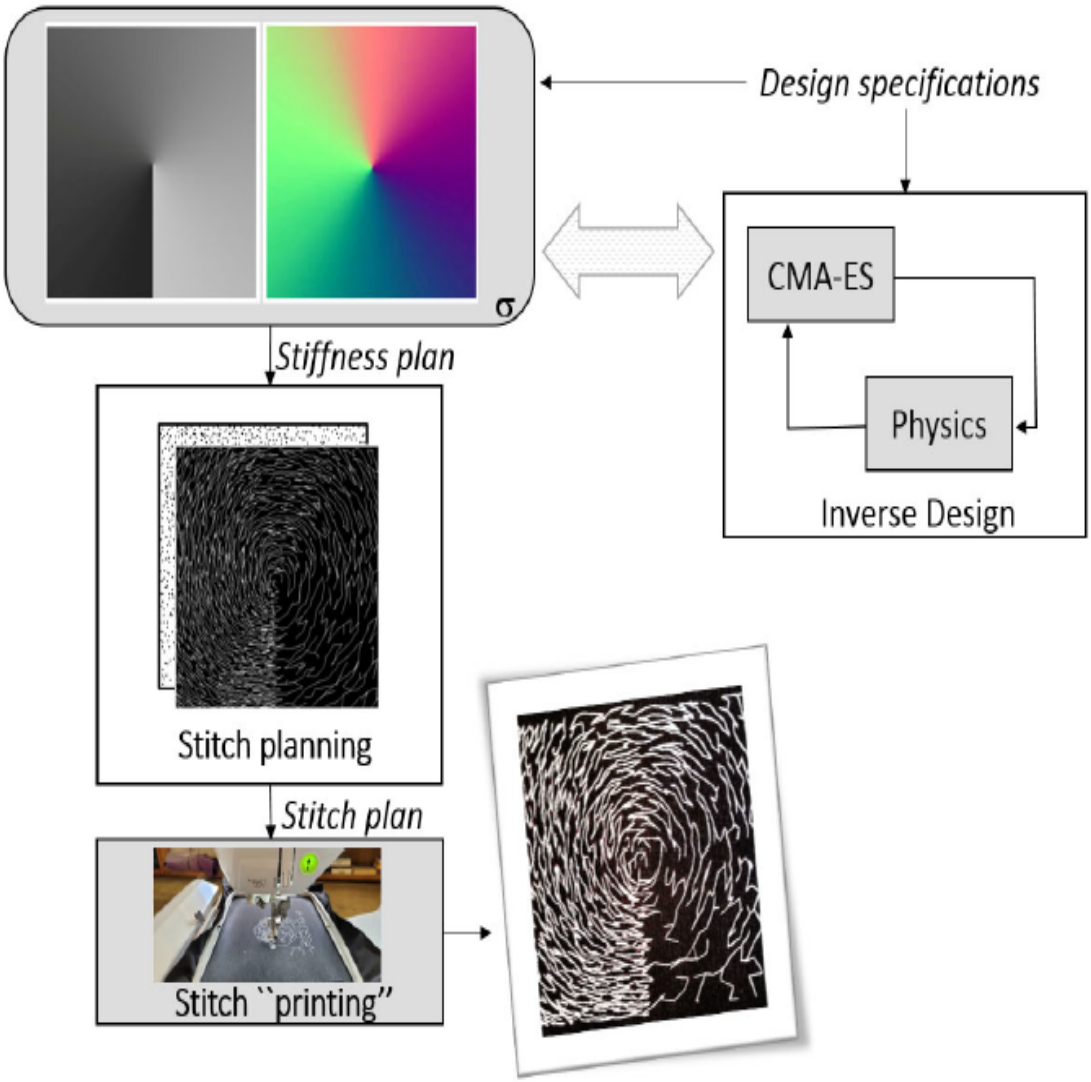

Figure 3.1: Overview of the approach

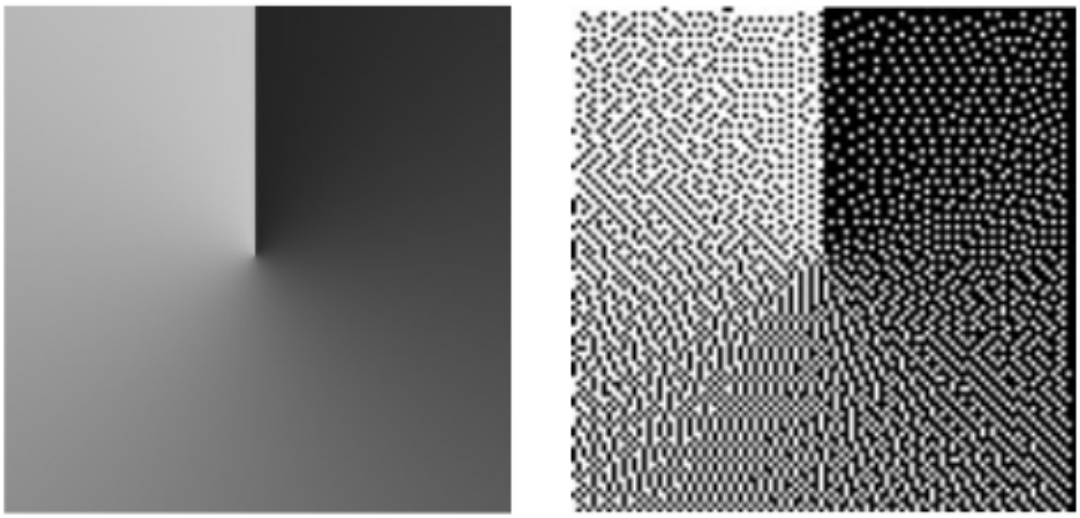

Figure 3.2: A sample density map and its corresponding Floyd Steinberg dithered image

are interpreted as being more dense than others. The denser a region, the more points sampled in that region with the effect of stitching being highest. They are regions that will be influenced the most by the embroidery made on top of them, because more points equates to more stitches.

The simplest way to sample points is by dithering the input density map. Dithering with exactly two colors - black and white (which converts the image to a bitmap image) produces the image seen in Fig 3.3. We make use of the Floyd Steinberg dithering algorithm for this example [32]. In trying to approximate the best representation of the image with just 2 colors, the algorithm produces more black pixels in regions with higher density. We interpret these black pixels as the vertices for the stitch planner.

The dithering approach to sampling is susceptible to aliasing artifacts (Fig 3.4). All points sampled end up being mapped to the centre of a pixel in the density map image, which leads to the formation of straight or diagonal lines when these points are joined together by the stitch planner (Fig 3.4). Some other patterns like the reverse-L shaped staircase can be seen in the less dense regions.

To solve this problem, we employ an approach that adaptively samples points in a region based on the average density requirements of that region. We try to avoid any kind of bias by sampling point(s) in a random location within a pixel. The density map grid (100X100, since that's the max our embroidery machine can support) is divided into a coarser one (eg. 10X10 - we call this a sub-grid), average the intensity values in the sub-grid, and determine based on that, how to scale the sub-grid. So a scale of 1.0 (which preserves the original size of 10X10 - 100 jittered points) would be ideal for a region that is completely black (intensity 0), and a scale value of 0.7 (would turn into 7X7 now - 49 jittered points) would be ideal for a region that is grey (intensity 128). We compute the sub-grid size after scaling as follows,

$$Let\ p = \frac{(255 - a)}{255} \tag{3.1}$$

We want

$$s \leq pg^2 \implies s = \lfloor g\sqrt{p} \rfloor \tag{3.2}$$

s represents the sub-grid size after scaling, g is the sub-grid size before scaling, and a represents the average intensity of the region of interest. We can extend the same idea to cram in even more points into a single pixel, by modifying the formula as follows,

$$s = k \cdot \lfloor g\sqrt{p} \rfloor \tag{3.3}$$

k is referred to as the inflation constant. We set $k = 1$ in the example in Fig 3.4. It can be set to values larger than 1 for higher density requirements.

The adaptive sampling approach solves the aliasing issue, but has a tendency to sample points in clusters too close to each other some times. When the stitch planner connects these points by making edges between them, some edges end up crossing over each other. Ideally, we want to sample points in a uniformly random fashion keeping the relative spacing between them equal. We turn to using the Centroidal Voronoi Tesselation (CVT) technique to sample points [8]. Sampling

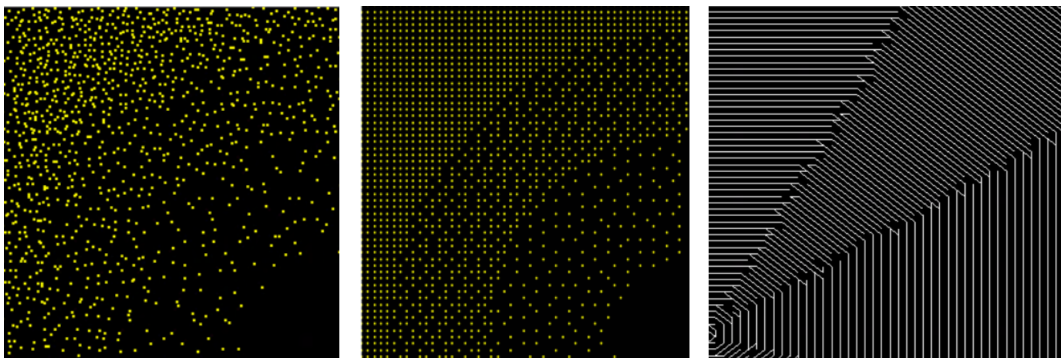

Figure 3.3: Left: Adaptive sampling. Centre: Sampling with FS dithering. Right: Aliasing artifacts produced by FS dithering. Diagonal stitches are closer to each other than the vertical and horizontal ones.

with CVT produces stitch plans where consecutive distance between stitches in a region stays roughly the same, which helps avoid stitch crossings (Fig 3.5 and Fig 3.6). CVT can be formally described as follows,

Given a set of voronoi regions $\{V_i\}_{i=1}^{k}$, the centroid of the region c_i is computed using a weighted average of the density values for the voronoi vertices given by $\rho(x)$, where x is a voronoi vertex forming a voronoi region of interest.

$$c_i = \frac{\int_{V_i} x\rho(x)\,dx}{\int_{V_i} \rho(x)\,dx} \tag{3.4}$$

For k generators $\{z_i\}_{i=1}^{k}$, we need to ensure $c_i = z_i$, for all k, which means all of our sampled points need to be close to the centre of mass of the voronoi regions. The density function $\rho(x)$ can be computed from the pixel values of the density map. There has been considerable progress in making algorithms that generate CVT's quickly and efficiently [18, 12]. We employ a simple implementation of Lloyd's algorithm to generate the final CVT [7]. We stop after $\|c_i - z_i\|^2$ becomes sufficiently small.

3.1.2 Path Planning

The path planner developed by [24] produces a path as output. In conventional graph theory, a path touches every vertex exactly once. This decision was made because the embroidery machine used doesn't have the ability to jump from one point to another on the fabric without making a stitch. On the other hand, the planner we developed produces a tree as output. A tree

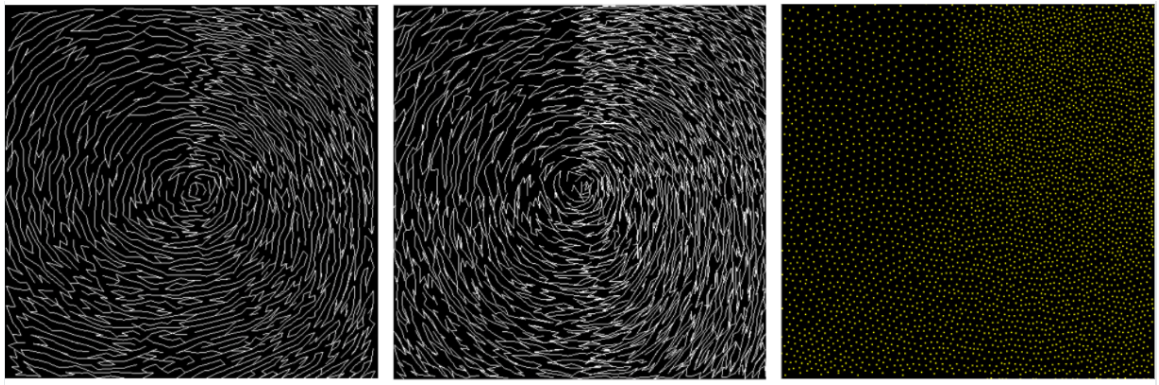

Figure 3.4: Left and Centre: Stitch plans generated after sampling with CVT and Adaptive Stratified Sampling respectively. Right: Sampled points generated using CVT

has several advantages over a path as we will see later. Since the embroidery machine can't make a tree, we convert it into an Euler walk (a path where a vertex or edge can be used more than once). A simple depth first traversal can be performed on the tree to produce a walk. This also means that we end up making a stitch between two points twice - once while going down in the search (exploration phase) and once while going up (during the unwinding of the recursion). This has the positive consequence of making individual stitches stronger, since they have a tendency to break under greater loads.

3.1.3 Cost functions and Material Properties

Before we take a look at the core planning algorithms, it is important to understand the role of cost functions and the physical changes they bring about. The previous work [24] was solely concerned with developing a planner that could infuse isotropic changes to the cloth in varying degrees across the fabric surface. We offer a solution that can inject isotropic as well as anisotropic properties in the fabric and also mix the two if required. We will now describe the differences between the two different types of stitching.

Isotropic or uniform stitching enables equal stretching in all directions at any given point on the fabric. Anisotropic stitching enables non uniform stretching in different directions at any given point on the fabric. Fig 3.6 shows these concepts in pictorial form. A good balance between horizontal and vertical stitches gives us uniform stiffening, since there is no real bias in stitch direction

(Fig 3.6). We choose a cost function that promotes 90 degree turns between stitches. The following cost function does the job for us,

$$cost(u, v, w) = -\alpha^{-|w-v|}\beta^{-|(v-u)\cdot(w-v)|} \tag{3.5}$$

(v, w) is the stitch we want to make, with *(u, v)* being the predecessor stitch to *(v, w)*. As can be seen in the cost function, these two stitches forming a straight line or close to forming one will be penalized. α and β control the importance of the two cost terms.

For anisotropic stitching, we introduce the concept of a direction or normal map (Section 3.1.4). This map is an RGB image that encodes a vector field which is used to lookup preferred stitch direction at a given point on the fabric (Fig 3.7). For most cases where a normal map is employed, we get some sort of bias in stitch direction, giving us non uniform or anisotropic properties (stretch along all directions is not the same at every point). The following cost function is employed to produce this effect,

$$cost(u, v, n) = -\alpha^{-|v-u|}\beta^{|(v-u)\cdot n|} \tag{3.6}$$

(u, v) is the stitch we want to make, and n is the preferred direction at point u, computed from the normal map. We flip the sign of the exponent on β this time, to encourage *(u, v)* to align with n. Results can be seen in Figure 6 (left).

3.1.4 Normal maps

We make use of normal maps to encode directional information [3]. Every point we sample on the fabric gets mapped to a direction that needs to be followed at that point, or the direction a stitch made from that point needs to head toward. Note that this is just a preference, and the planner may not always be in a position to follow the preferred direction exactly as is. We use the following formulas to convert RGB values to 2D direction vectors.

Direction vector $\vec{n}$ is given by,

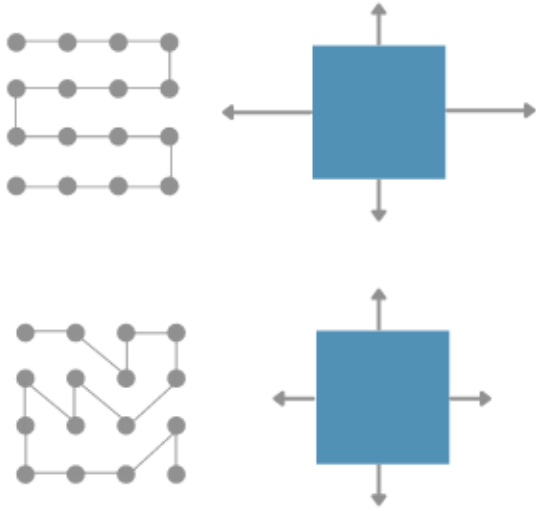

Figure 3.5: Top: An example of an anisotropic pattern. Bottom: An example of an isotropic pattern. The length of the arrows denote magnitude of stiffness along the direction pointed to by the arrow

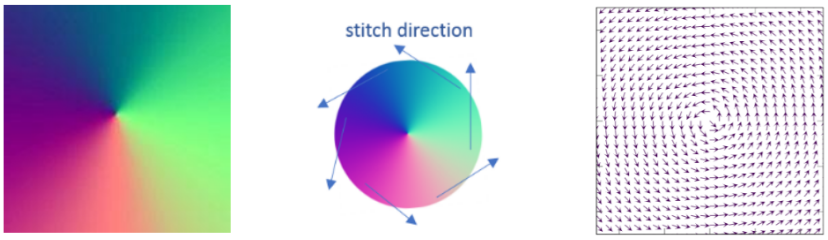

Figure 3.6: Left: RGB image representation of a normal map. Centre: A rough estimate of stitch directions. Right: Normal map visualized as a vector field.

$$\vec{n} = \begin{pmatrix} nx \\ ny \end{pmatrix}$$

$$nx = \frac{2 \cdot r}{255} - 1, \;\; ny = \frac{2 \cdot g}{255} - 1$$

r and g represent the red and green channel values respectively. This formula maps pixel values [0, 255] to direction values [-1, 1]. Note that these formulas do not make use of the blue color channel.

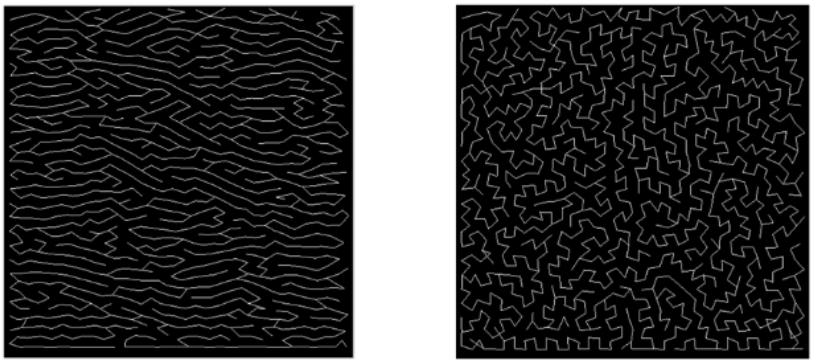

Figure 3.7: Left: Stitch plan produces anisotropic stiffening. All stitches are biased to be horizontal, Right: Stitch plan produces isotropic stiffening. Equal balance of stitches going in all directions, totally unbiased

We will refer to these directions as normals in the future. Figure 3.7 shows a visualization of how the normals look for the given normal map.

3.1.5 Planning with TSP

In this section, we setup the path planning problem as a typical Travelling Salesman Problem, where every sampled point on the fabric needs to be visited exactly once. Specifically, we opt for the greedy TSP technique based on the work of [4] as the previous authors did [24]. In the greedy TSP algorithm, we try to find a "good enough" solution, effectively making it an approximate algorithm (1.5 approximate to be precise) since finding an exact solution for tens of thousands of vertices isn't feasible [14].

We start by picking a vertex at random, and try to find potential candidate vertices that can be used to make a stitch. The quality of these connections is evaluated using the cost functions described in Section 3.1.3. We pick the vertex for which the cost value is the least at every iteration, and repeat this process until we run out of vertices to visit. Figure 3.9 shows a stitch plan produced with this method.

Because we are limited to only making a path, this strategy tends to produce stitch plans with quite a few undesirable long stitches, which we also call jump stitches. Long stitches aren't desirable for two reasons; one - the embroidery machine we employ is limited to making stitches at most 15 mm long and two - long stitches break more easily since majority of the load is at the

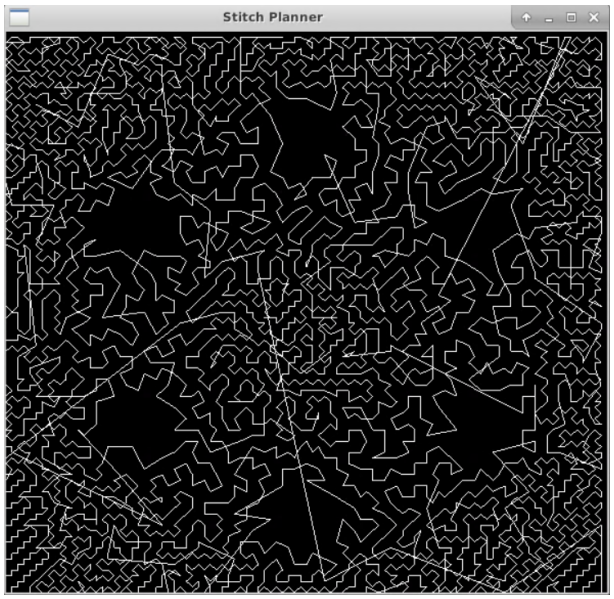

Figure 3.8: Greedy TSP without 2-opt (notice the long jump stitches)

centre of the stitch when it is pulled outwards. We introduce another technique to deal with long stitches - the 2-opt algorithm. The core idea behind the 2-opt technique is if there are two pairs of edges crossing each other in a path, the algorithm will swap them so as to get rid of the crossing. A jump stitch is a crossing that can be resolved using this approach. The result in Figure 3.9 is post-processed with the 2-opt algorithm to produce the result in Figure 3.10

The examples in Figures 3.9 and 3.10 produce fully isotropic cloth samples. A post-processing solution like 2-opt is hence useful to resolve jump stitches, since the resulting swap doesn't bias stitches to be in a certain direction. This strategy of path planning doesn't generalize very well to anisotropic stitching. Figure 3.11 demonstrates this, as multiple violations of the desired stitch direction can be seen on the fabric surface which can affect the overall quality of the samples we physically manufacture. Resolving these issues will be our primary motivation for coming up with the techniques described in the next section.

3.1.6 Planning with Shortest Path Trees

As we saw in the previous section, the TSP approach has a tendency to generate off-directional and long stitches (in the case where no 2-opt is employed) we would like to avoid, even if there are only a few of them (Figure 3.11). These stitches are difficult to resolve without adding

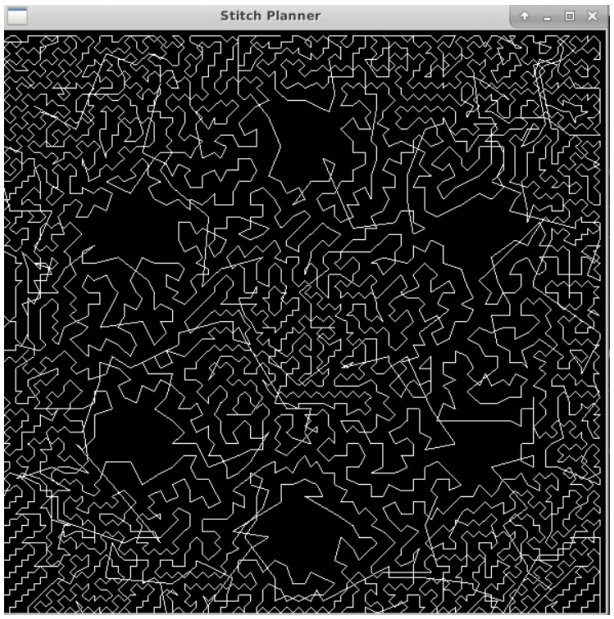

Figure 3.9: Greedy TSP with 2-opt

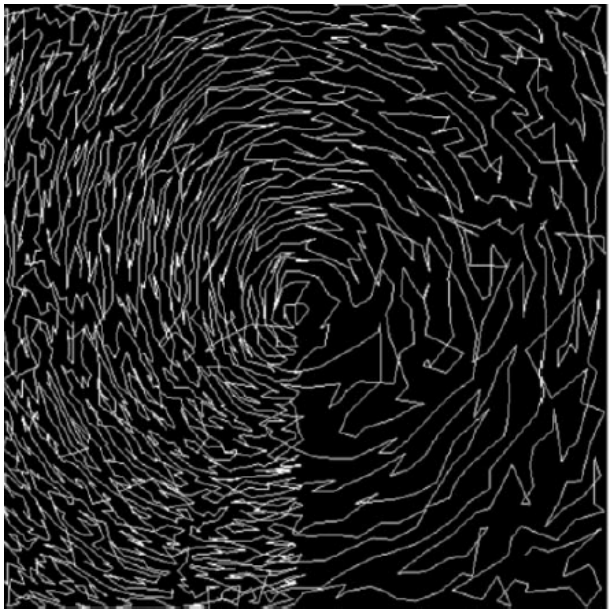

Figure 3.10: Greedy TSP applied to solve anisotropic stitching

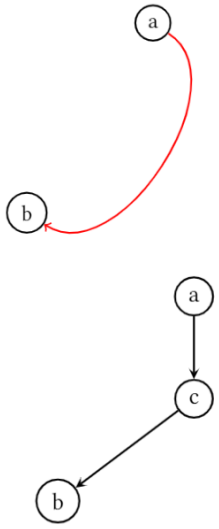

Figure 3.11: Example paths between two vertices a and b

an extra degree of freedom - the freedom to stop at a point and continue exploration from another, a kind of flexibility a path doesn't offer. The best data structure for this purpose is a *graph*. We continue using cost functions from section 3.1.3 to define the weight of an edge in our graph.

We would like to generate a graph that is minimal in it's structure, and one that covers all sampled points on the fabric, whilst keeping stitches directional and reasonably short. From the description provided, opting for a Minimum Spanning Tree (MST) [5] approach seems ideal. We go with a slightly different variant that goes by the name of Shortest Path Tree (SPT), for reasons that will be elaborated later in the section. Our problem can then be formulated as a classic single source shortest paths search, which can be solved using Dijkstra or Bellman Ford [6, 2].

Let us first try to justify negative edge weights, and then move on to the proposed solution. In the example shown in Figure 3.12, there are two different paths going from vertex 'a' to vertex 'b'. Let's assume that the black colored path (composed of exactly two stitches) is shorter than the red one. The red path might be longer in length, but if it satisfies directionality better than the black one, we've got to pick it. Now, if the edge weights were positive, we would most likely opt for the black path - it will be extremely hard to penalize stitches (a, c) and (c, b) with big positive numbers so as to not pick the route from 'a' to 'b'. This is where negative edge weights prove to be useful, since they let us do the following - create arbitrarily long stitch paths that follow direction extremely well, whilst doing so in a more "continuous" fashion, which gives our stitch plans more

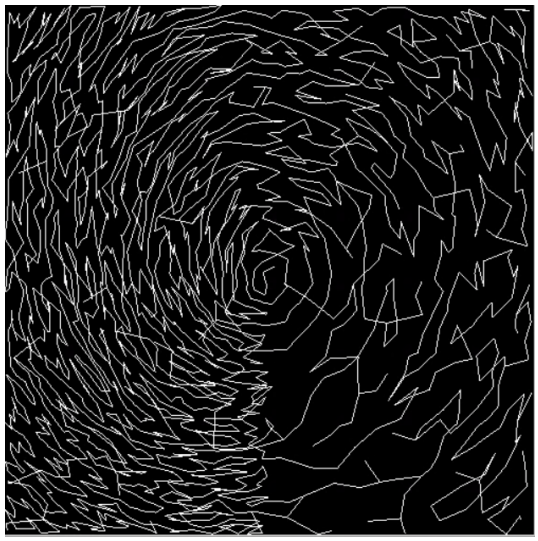

Figure 3.12: The stitch plan rendered with Dijkstra's algorithm.

of a path like feel and reduce the possibility of making too many small "branch stitches" - singular stitches that poke out to form leaf nodes (these can be seen in the low density regions of our stitch plans). These requirements point towards a greedy search, so we choose Dijkstra's algorithm to generate the SPT. The difference between Dijkstra's algorithm and ours can be resolved with an additional condition check in the relaxation step, where we do not update the cost of a node if it is already a part of the SPT we are building (since the cost will only get smaller due to negative edge weights). We opt for an even simpler approach - filtering out nodes that are part of the SPT from the set of neighbor nodes we would like to relax in every iteration.

The algorithm does produce a reasonable looking stitch plan where both direction is followed and stitch length is kept in check, while faking a continuous looking stitch plan. It does however, produce some off-directional stitches, as can be seen in Figure 3.13. This isn't a shortcoming of our approach. A shortest path may have certain stitches that might go off-direction, since the cost function is dependent on both the length and direction of the stitches.

The stitch plan being a tree is something we will take advantage of to get over this issue. Unlike paths, a tree isn't very sensitive to the removal/addition of a single edge. The core idea is simple - remove a *bad (off-directional)* stitch and replace it with a *good* one. It is important to note we perform this post-processing clean-up only when fully anisotropic solutions need to be produced, since that is when directionality requirements are the most stringent.

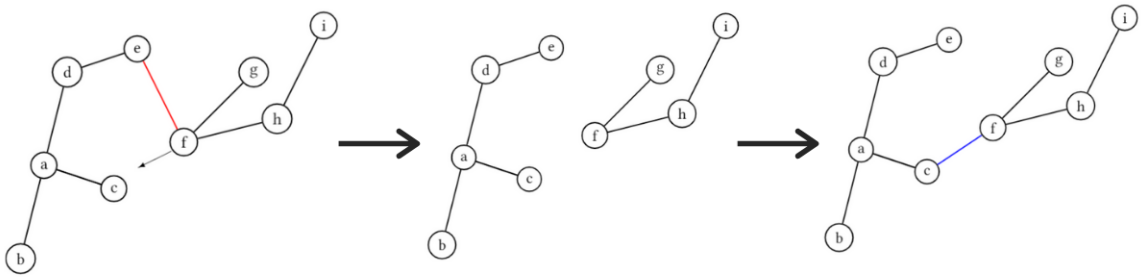

Figure 3.13: A typical iteration of the cleanup procedure

Consider the example shown in Figure 3.14. In the leftmost graph, the arrow at 'f' represents the preferred direction. The stitch (f, e) goes orthogonal to this direction and will be flagged as an off-directional stitch. If we were to remove stitch (f, e), we would be left with 2 connected components, as can be seen in the centre diagram. This is because a tree does not contain any cycles and connects all nodes to each other without any extraneous edges. Each and every edge hence, becomes important for connectivity. The only way to restore connectivity is to make an edge from any node in the first connected component to any other in the second. We make a stitch/edge between the two components that gives us the least cost (refer to the *getBestNode()* procedure in Algorithm 2). One of the sub-problems we need to solve to achieve this, involves labelling connected components. There exist a variety of algorithms for connected component labelling [13]. Luckily, we have only two of them at any given point in time, so we opt for the simplest one - a depth first traversal. Results of the cleanup procedure can be seen in Figure 3.15 (right).

We also design a simple test to gauge the efficacy of the two proposed solutions for non-uniform stitching. We generate a stitch plan with one of the two algorithms, and reverse the directions of the stitches to get back the colors in the normal map to produce a "reversed map". The closer this reversed map is to the original normal map, the better. The colored pixels in this reversed map correspond loosely to the sampled points on the fabric. We compute these pixel values by comparing the similarity of the stitch direction vector at a given point with the preferred direction

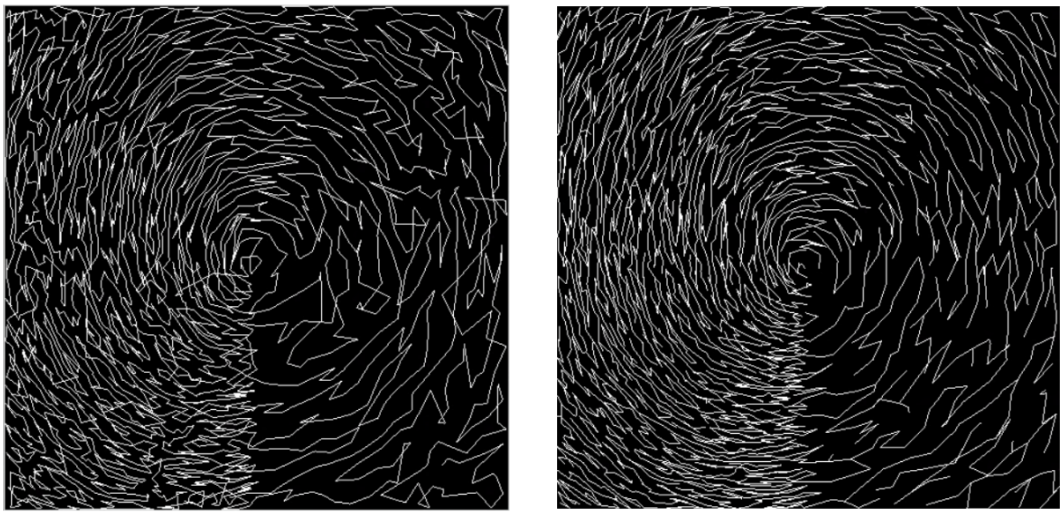

Figure 3.14: Left: Stitch plan rendered with modified TSP. Right: Stitch plan rendered with Dijkstra's algorithm and the post-processing cleanup routine.

in the normal map corresponding to that point, and use that result (ranging from 0.0 to 1.0) as a scale factor to multiply the corresponding color in the normal map to get the final color in the reversed map. A single point might have multiple stitches going out from it (since we have a tree now) and these will have to be averaged out to get a final direction vector for that point. Examples of these maps can be seen in Figure 3.16.

Algorithm 1 Stitch Planner

```
1: procedure PLAN(start, points, normal)
2:     // Run Dijkstra's to generate the SPT
3:     spt ← generateSPT(start, points, normal)
4:     stitches ← offDirection(spt)
5:     // fix the off-directional stitches
6:     cleanup(spt, stitches, normal)
```

3.1.7 SPT Implementation

In our Dijkstra implementation, we start from a random vertex as the source, sample neighbors around it on a user-defined radius set to 5 units typically, and insert them into a priority queue by assigning each of them an appropriate cost value from the cost functions described above. The lowest cost vertex is chosen at the start of every iteration for processing, followed by the relaxation of it's neighbors if need be. The process terminates when there are no more vertices left to pro-

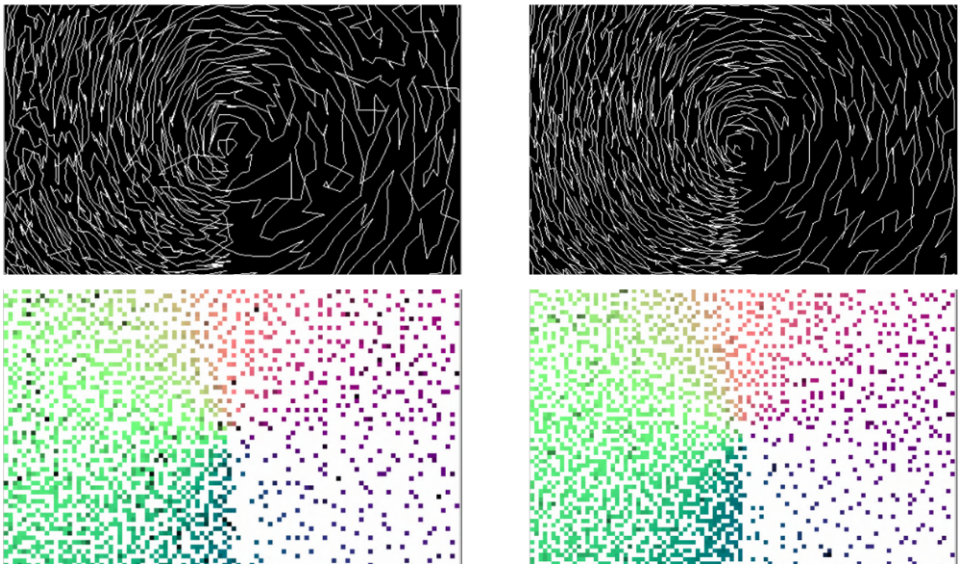

Figure 3.15: Top: Stitch plan. Bottom: Reversed Map of normal map from Figure 3. Left: Stitch plan rendered with TSP. Right: Stitch plan rendered with Dijkstra's. The reversed map on the left has more dark pixels than the one on the right.

cess. While other approaches such as best first search can be used to generate a tree, our Dijkstra approach produces a more continuous looking stitch plan and avoids making too many branches.

3.1.8 No go boundaries

The result produced in Figure 3.17 induces some inconsistencies in the stitch plan on the circular regions. Ideally, the circular regions should have more or less the same patterns woven over the area they occupy on the fabric. However, when observed closely, we find out that the two circles at the top and one at the center have stitches crossing over their centres, an artifact not seen in the bottom two circles. This leads to the formation of soft spots at the centres of the two circles at the bottom - it's easy to push the fabric up, if a force were applied to it. The remaining three circles on the other hand, would resist any kind of force applied to their centres due to the stiffness introduced by the stitches going through them. We try to enforce certain constraints the planner has to adhere to, and propose the usage of a "no-go boundary" in order to obtain consistent results. A no-go boundary/boundaries define a set of segments on the fabric, that no stitch can cross over. Figure 3.18 shows an example where we specify no-go boundaries visually over a normal map. This

Algorithm 2 Cleaning up off-directional stitches

```
1:  procedure CLEANUP(spt, stitches, normal)
2:      for all e ∈ stitches do
3:          u ← e.u
4:          v ← e.v
5:          spt.remove(e)
6:          // Label the connected components with numbers 1 and 2
7:          spt.dfs(u, 1)
8:          spt.dfs(v, 2)
9:          // Select best from other connected component
10:         ub ← getBestNode(u, c1, 1)
11:         vb ← getBestNode(v, c2, 2)
12:         if c1 < c2 then
13:             nu ← u
14:             nv ← ub
15:         else
16:             nu ← v
17:             nv ← vb
18:         // Compare costs between the old and new edge
19:         w1 ← cost(nu, nv, normal[nu])
20:         w2 ← cost(u, v, normal[u])
21:         if w1 < w2 then
22:             spt.add(edge(nu, nv))
23:         else
24:             spt.add(edge(u, v))
```

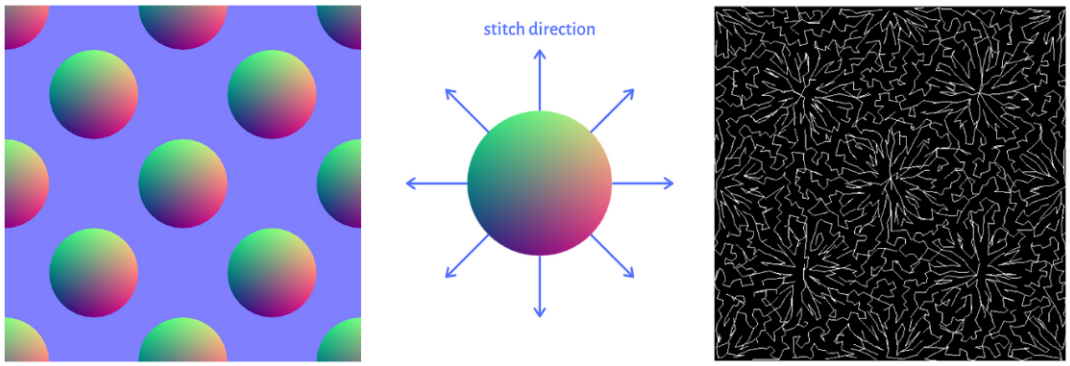

Figure 3.16: Each circle in the normal map behaves in the opposite way as the one in Figure 3. A typical direction goes normal to the surface of the circle, in contrast to the tangential direction we saw before. The blue region does not represent any specific stitch directions, indicating uniform stiffening.

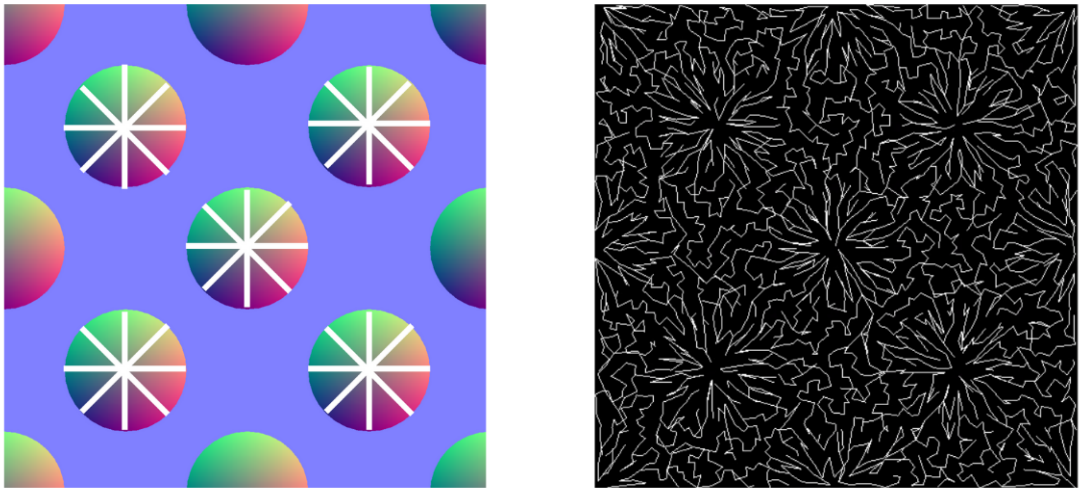

Figure 3.17: Enforcement of no-go boundaries yields results that are consistent throughout the stitch plan

is a semi-automatic approach to solving the problem, since these segments need to be chosen by the user via a line/segment drawing interface. We can guarantee that there wouldn't be any stitches going over the centres of the circles, producing soft spots in each of the circle centres.

3.1.9 Mixing Material properties and cost functions

Another advantage of using a tree structure to produce stitch plans is the generalization capability it offers. We can seamlessly move from complete anisotropy to complete isotropy by changing a single number ω. We use ω as an interpolation constant that can be dialed from 0.0 to 1.0 - fully anisotropic to fully isotropic respectively. Mathematically, this can be represented as follows,

$$cost_{total} = (1 - \omega) \cdot cost_1 + \omega \cdot cost_2. \tag{3.7}$$

where $cost_1$ and $cost_2$ are given by equations 3.5 and 3.4 respectively. The new edge weight will then be given by $cost_{total}$

Figure 3.19 shows some results when applying this formula to modify the cost functions

Figure 3.18: Maps used to generate stitch plans shown in Figure 3.19

for maps described in Figure 3.19. The pink normal map is interpreted as horizontal stitching everywhere. The density map used is flat (no gradient change) and sampled using stratified uniform sampling.

3.2 Map optimization with Inverse Design

As we saw in the previous sections, the density and normal map form the main constraints our stitch planning algorithms need to adhere to. The density map specifies how many stitches need to be made in local regions of the fabric and the normal map specifies the direction those stitches need to head toward. These maps can become increasingly tricky to design by hand, since the resulting outcome they have on the stitch plan being made and how they might affect different properties of the cloth are very hard to predict. In this section, we will look at techniques for automating the process of generating these maps based on a user defined performance criteria. The user of the system can specify external forces being applied on the cloth and the desired deformation of the cloth in response to those forces. An optimization algorithm then computes density and normal maps that give a solution producing the target cloth behaviour. At the heart of this optimization algorithm is a cloth simulator, which helps us predict how different estimations of the density and normal map

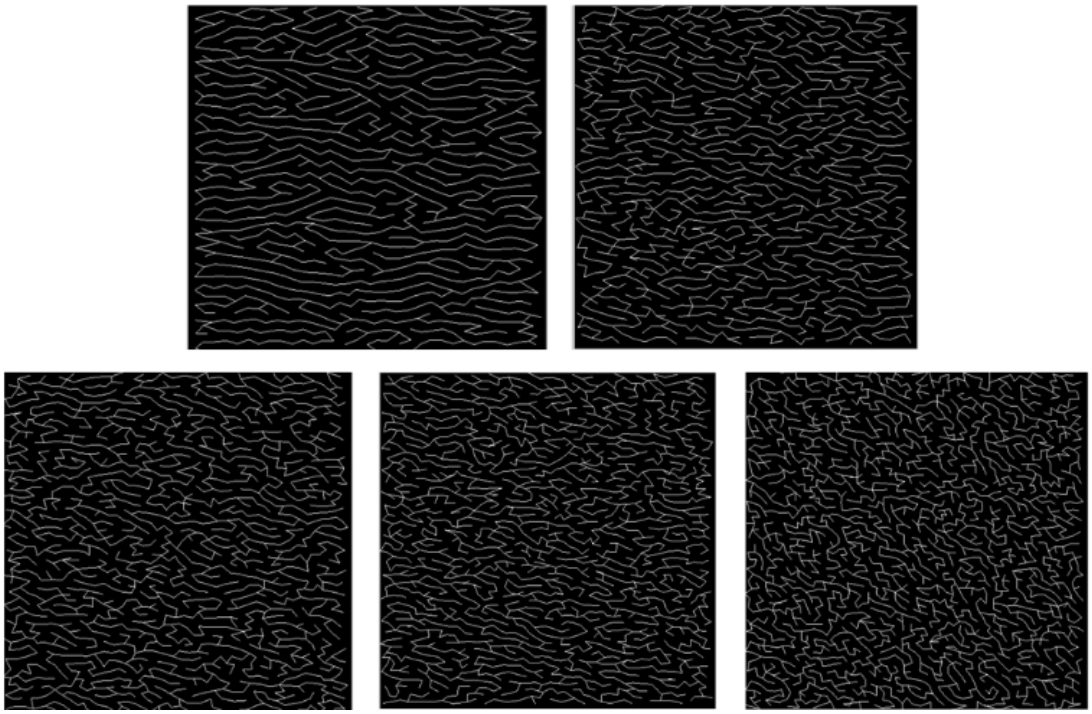

Figure 3.19: ω values ranging from 0.0 to 1.0. Top left - 0.0 (completely anisotropic), Top right - 0.3, Bottom Left - 0.5, Bottom Centre - 0.7, Bottom Right - 1.0 (completely isotropic)

affect the cloth. These estimates are iteratively tuned until the target performance on the cloth is achieved. In the following sections, we will dive into the details of how this is accomplished.

3.2.1 Cloth Simulator

We employ a spring mass based cloth simulator for our experiments, although the optimization technique we describe can generalize to other simulators as well by tweaking how the density and normal maps are mapped to the cloth surface. For example, when mapping onto a spring mass system, we change the stiffness constants of the springs that make up the cloth surface. If we were to use a continuum based model instead, we would alter energy functions associated with different condition functions for stretching, shearing and bending as described in [1].

The simulated cloth is a 2D lattice of points (can be thought of as a 2D grid), with each point/vertex connected to its 8 nearest neighbors with a spring. The internal forces on a spring in our system are governed by Hooke's law. The force experienced by a vertex i from the influence of vertex j in this system is given by:

$$f_i(x) = \frac{-k(\|x_i - x_j\| - L)(x_i - x_j)}{(\|x_i - x_j\|)}$$

x_i, x_j are the positions of vertices i and j respectively. L is the rest length (the length before simulation starts) and k is the *spring stiffness constant*. As mentioned earlier, the spring stiffness constant is the most important thing we care about for optimization. By estimating the right values for the spring stiffness for every spring in the system, we can make the cloth behave in interesting ways, and eventually approximate a pose that matches our target.

The 2D lattice we use for the experiments has dimensions of 30X30. For an nXn dimensional structure we will have exactly $4n^2 - 6n + 2$ springs or roughly of the order n^2. Estimating a different spring coefficient for so many springs is extremely difficult. This is the reason we turn to using basis functions to approximate the density and normal maps and devise a scheme to map them to the springs to estimate spring values.

3.2.2 Basis functions

A density map is a grayscale image that can be parameterized or decomposed in the frequency domain using many different techniques (Wavelets, DFT, DCT, etc.). We can convert these frequency representations to spatial ones easily to get most (if not all) grayscale images. We make

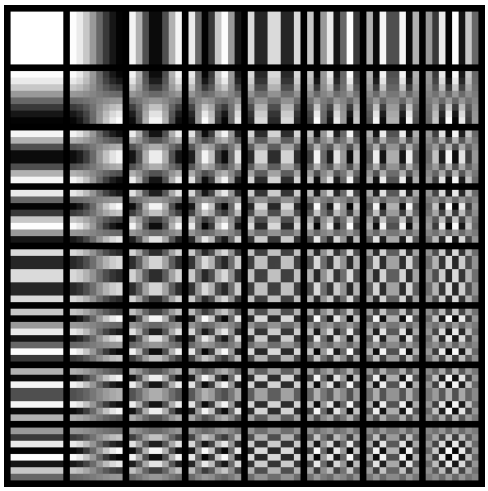

Figure 3.20: DCT basis functions from x = 0, y = 0 (no gradient change in both directions) to x = 8, y = 8 (every consecutive pixel has a different color, maximum gradient change for an 8X8 image)

use of the Discrete Cosine Transform representation for this task [9], a frequency decomposition technique used heavily in JPEG image compression (Fig 2). The core idea is to break an image down into a series of low and high frequency components and eliminate most of the high frequency components, since the contributions made by them are more often than not insignificant. Each basis function can be thought of as an image, and have a weight/parameter associated with it, which signifies the contribution of that image to the final density map. These images are then combined together linearly to produce the density map which is then mapped onto the lattice representing the cloth in the simulator.

We apply the same procedure described above to normal maps as well. A normal map is an RGB image specifying a vector field for stitch direction. We use the same basis functions three times for normal maps - one for red, one for green and one for blue.

Let $\mathcal{D} \in [0, 255]^{k \times k}$ be the greyscale density map that we would like to generate. We use the Discrete Cosine Transform (DCT) and express $\mathcal{D}$ as a weighted sum of $k \times k$ sinusoidal basis images (functions) with weights $\mathbf{w}^{\mathcal{D}} \in R^{k \times k}$ as follows:

$\mathcal{D} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} w_{ij}^{\mathcal{D}} \mathbf{B}_{ij}$, where $\mathbf{B}_{ij} = \cos(\frac{(2x+1)i\pi}{2k}) \cos(\frac{(2y+1)j\pi}{2k})$.

$\mathbf{B}_{ij}$ denotes the $(i \cdot k + j)^{\text{th}}$ basis image where $0 \leq x, y < k$, and $\mathbf{w}_{ij}^{\mathcal{D}} \in R$ is its corresponding weight.

We employ a similar decomposition approach for the RGB normal map $\mathcal{N} \in [0, 255]^{3 \times k \times k}$,

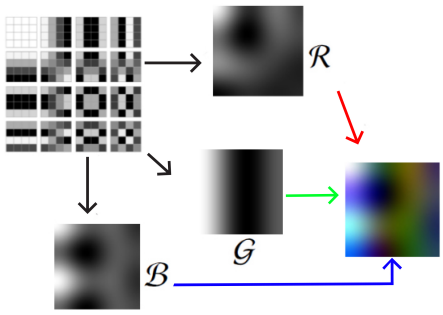

Figure 3.21: A weighted combination of these leads to a variety of 4×4 greyscale images. 3 greyscale images can be combined together to form a normal map

where we treat each channel as a greyscale image. The weights of the corresponding bases are $\mathbf{w}^{\mathcal{R}}, \mathbf{w}^{\mathcal{G}}, \mathbf{w}^{\mathcal{B}} \in R^{k \times k}$ for the red, green, and blue channel, respectively.

The DCT decomposition allows for a more confined and structured space of input maps as compared to naively searching for pixel values of the corresponding maps.

3.2.3 Defining Performance Criteria

We define an error function to be equal to the sum of squared differences between the estimated positions of a point on the cloth at equilibrium and the target/user defined position of the corresponding point. The smaller this value, the better. Our objective hence, is to estimate parameters that minimize this error function.

$$E_{target} = min \ \ \sum_i (x_i - x_i^{'})^2$$

$x_i, x_i^{'} \in R^3$. x_i - expected position, $x_i^{'}$ - predicted position. It is important to note that these positions can be for the cloth or for an object resting on a cloth for coupled simulations. The later sections will make this idea more clear.

In summary, our design problem is defined by the following components:

- A physical system with m points having positions $\mathbf{x} \in R^{2m}$.
- User-defined target criteria expressed as a function $E_{\text{target}} : \mathbf{x} \rightarrow R$.
- An unknown greyscale density map $\mathcal{D} \in [0, 255]^{k \times k}$ expressed as a weighted combination of $k \times k$ basis functions.

- An unknown normal map $\mathcal{N} \in [0, 255]^{3\times k\times k}$ expressed as a weighted combination of $k \times k$ basis functions.

3.2.4 Embroidered Cloth Simulation and Optimization

In order to estimate the stitch plan within the optimization cycle, we modify our cloth simulator to account for the impact of the embroidery on the local tensile strength of the fabric. We can incorporate stretch and shear forces within the simulation, computed through Hookean springs that connect the node masses. The physical system consists of m particles organized in a quadrilateral mesh having positions $\mathbf{x} \in R^{2m}$, velocities $\mathbf{v} \in R^{2m}$, and inertia matrix $\mathbf{M}$. The cloth is constrained in the examples we present in various configurations along the boundary. It also experiences a gravitational force. We couple the cloth with a rigid body simulator with penalty forces that account for self-collisions and inter-penetrations. The system evolves through a discrete set of time samples with constant time step h according to Newton's laws of motion.

The goal of our system is to obtain a user-specified target design. We formulate this goal as an optimization problem

$$\begin{aligned} \underset{\mathbf{w}^{\mathcal{D}},\mathbf{w}^{\mathcal{R}},\mathbf{w}^{\mathcal{G}},\mathbf{w}^{\mathcal{B}}}{\text{minimize}} \quad & E_{\text{target}} + \theta_{\text{freq}} E_{\text{freq}} + \theta_{\text{stitches}} E_{\text{stitches}} \\ \text{subject to} \quad & \text{static equilibrium,} \end{aligned} \tag{3.8}$$

where static equilibrium is expressed as the condition of the particles being at rest (average particle speed is less than a small ϵ away from 0), and $E_{\text{target}} : \mathbf{x} \to R$ denotes user-specified target criteria expressed at the equilibrium state (see below for specific examples). E_{freq} is an energy term that penalizes the use of high-frequency DCT basis functions as follows:

$$E_{\text{freq}} = (i + j + 1)\mathbf{w}_{ij} \tag{3.9}$$

The constant θ_{freq} defines the tradeoff between satisfying the target design and promoting more compact representations (lower frequency basis images). Finally, E_{stitches} denotes a regularization term that penalizes large number of stitches, with the weight θ_{stitches} controlling its importance. The fewer the number of stitches, the faster the embroidery machine prints the cloth sample. While there are different ways to capture this objective term, here we focus on decreasing the magnitude

of the density map weights:

$$E_{\text{stitches}} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} (\mathbf{w}_{ij}^{\mathcal{D}})^2 \tag{3.10}$$

We note that the DCT decomposition allows for a more confined and structured space of input maps as compared to seeking to minimize the objective in Equation 3.8 by directly searching for the pixel values of the maps. Given any set of weights, the main idea of our our inverse design formulation is to use the resulting density and normal maps to control the stiffness of the springs.

Let particles p and q be the endpoints a spring. According to Hooke's law, the spring potential is defined as:

$$\frac{1}{2}k(\|\mathbf{x}_q - \mathbf{x}_p\| - r)^2, \tag{3.11}$$

where $\mathbf{x}_p, \mathbf{x}_q$, denote the positions of the particles, $r \geq 0$ is the rest length, and k is the spring stiffness. We express the stiffness as $k = k_b + k_m$, where $k_b \geq 0$ denotes a minimum stiffness value, which is the same for all springs in the system, and k_m is the stiffness value inferred by combining the two input maps. In particular, given the density map $\mathcal{D}$, we first use bilinear warping to map $\mathbf{x}_p$ to $\mathcal{D}$ and infer a density value ρ_p using bilinear interpolation. After retrieving the red, green, and blue values of particle p from the corresponding $\mathcal{R}$, $\mathcal{G}$, and $\mathcal{B}$ maps, we use the equation described in Section 3.1.4 to determine the preferred direction $\mathbf{n}^p$. The contribution of p to the k_m stiffness value is then based on the computed density ρ^p and the degree of alignment of the spring's $(\mathbf{x}_q - \mathbf{x}_p)$ direction with $\mathbf{n}^p$, with less alignment leading to a stiffer spring:

$$k_m^p = \omega\rho^p + (1-\omega)\left|\frac{\mathbf{x}_q - \mathbf{x}_p)}{\|\mathbf{x}_q - \mathbf{x}_p\|} \cdot \mathbf{n}^p\right|, \tag{3.12}$$

where ω regulates the randomness as communicated by the blue channel in the normal map. We similarly compute the contribution of particle q to the k_m stiffness value and average the two to get the final k_m value.

Given the above mapping from input maps to spring coefficients our goal is to find the density map and normal map that meet the objective in Equation 3.8. We use CMA-ES (covariance matrix adaptation evolution strategy) [11], which is a derivative-free optimization approach, to solve the underlying optimization problem and find the set of weights $\mathbf{w} = \{\mathbf{w}^{\mathcal{D}}, \mathbf{w}^{\mathcal{R}}, \mathbf{w}^{\mathcal{G}}, \mathbf{w}^{\mathcal{B}}\}$. In our setup, we assume that the weights can be sampled from a multivariate Gaussian distribution with mean vector $\mu \in R^d$ and diagonal covariance $\sigma^2 I$, where $\sigma \in R_+^d$ and $d = (k \times k)^4$. At each CMA-ES

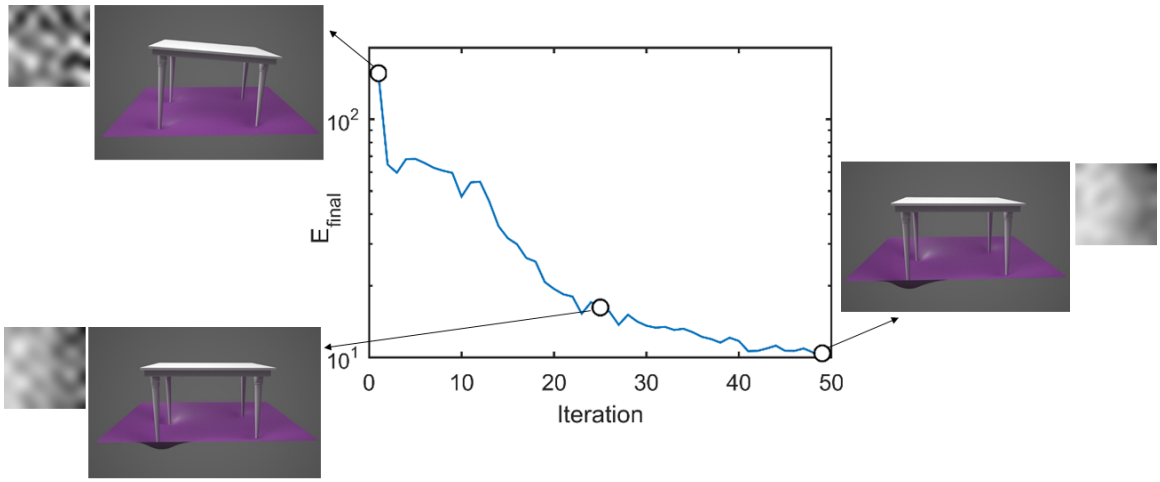

Figure 3.22: Evolution of the optimization process for the uneven table scenario shown in Figure 3.24. At each CMA-ES iteration, we plot the final objective value when the cloth reaches its equilibrium state.

iteration we generate n candidate solutions, i.e. n combinations of density map and RGB normal maps, by sampling n parameters from the Gaussian. We use each input map to modify the stiffness of the cloth and run the simulation until the the physical system reaches its static equilibrium. After assessing each solution according to E_{target}, we retain the top-λ% and fit a new diagonal Gaussian obtained from the covariance matrix by updating μ and σ based on the selected population.

3.2.5 Design Optimization

We showcase our proposed inverse design formulation on two problems, the uneven table scenario shown in Figure 3.24 and the cube scenario shown in Figure 3.23. Since each problem involves a coupled simulation between a rigid body and cloth, we specify the target design cost E_{target} (cf. Equation 3.8) as a function of the position of the rigid body at equilibrium.

We refer to Figure 3.22 for the convergence plot of CMA-ES in the table scenario. The goal here is to control the tensile properties of the cloth so that the table can stay balanced even though two of its legs are shorter than the other two. Geometrically, this means that the table top must be parallel to the surface of the cloth at static equilibrium. This is analogous to saying that the 4 corner points of the table need to be at the same height from the surface of the cloth. Mathematically, we

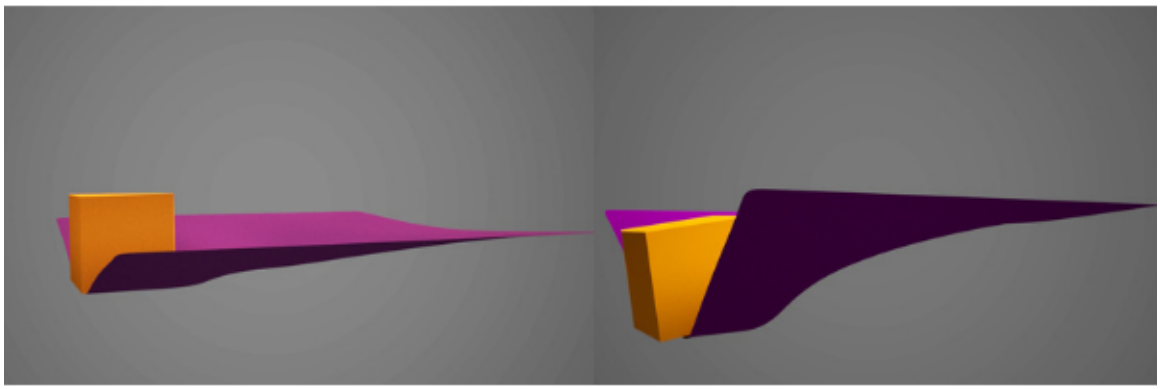

Figure 3.23: Inverse design - Cube. The goal is to ensure that the cube stays upright on the edge of the fabric. Cloth at equilibrium with (left) and without (right) optimization.

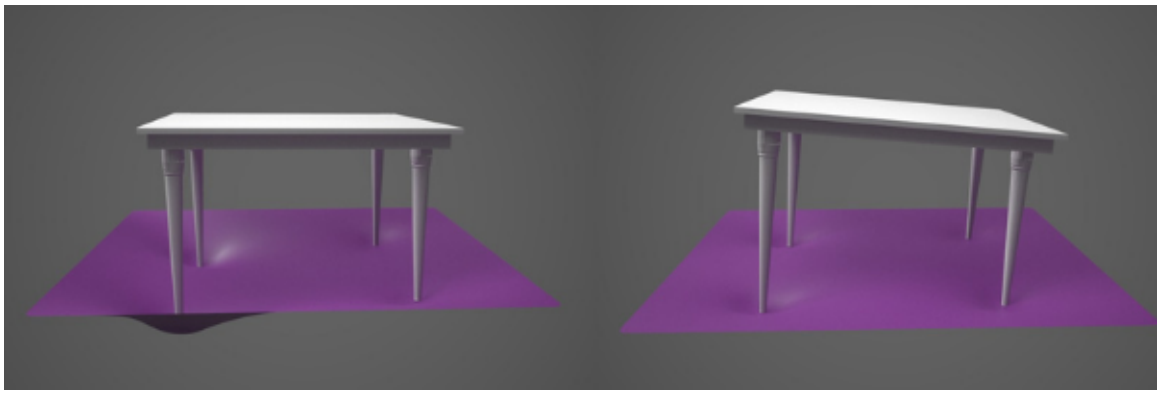

Figure 3.24: Inverse design - Uneven table. The goal is to control the stiffness of the fabric so that the table remains balanced. Cloth at equilibrium with (left) and without (right) optimization.

can describe this as follows:

$$E_{\text{target}}^{\text{table}} = \|x_1.y - x_4.y\|^2 + \|x_2.y - x_3.y\|^2 \tag{3.13}$$

where x_1, x_4 and x_2, x_3 are adjacent to each other and represent the 4 corners of our table top. As seen in the figure, CMA-ES quickly converges to a solution. After only 20 iterations, the target design criteria as defined above is already met but the algorithm continues in order to find a solution that will result in promoting low frequency basis and fewer stitches (compare the density map at 25 iterations and the one at 50 iterations where the smallest possible error is achieved).

Figure 3.25: Cube optimization - Density and normal maps produced by the inverse design optimization. The corresponding stitch plan is shown on the right.

Figure 3.26: Table optimization - Density and normal maps produced by the inverse design optimization. The corresponding stitch plan is shown on the right.

We repeat the same exercise for the second design optimization example, where the goal is to ensure that the cube sits sturdy and upright on the edge of the cloth and doesn't slide down (Fig 3.23). The target function for the chair remains the same as Equation 3.13. As shown in the figure, the CMA-ES optimization is able to generate an input stitch map that is very stiff at the very front edge of the cloth where the box gets in contact with the fabric (Figure 3.24), with the density being minimal in most of the other locations of the fabric. We note that such design criteria is hard to be attained without our proposed anisotropic stiffness model that provides a localized and precise way to control the stiffness of a fabric.

Chapter 4

Results and Validation

In this section, we will look at the physically manufactured versions of the digital results presented in the previous sections. We will also show the physical tests conducted to demonstrate the different anisotropic properties our algorithm was able to produce. The stress-strain curves of various stitch patterns will further prove the efficacy of our approach. The embroidery machine employed to fabricate stitch plans will also be explored briefly.

4.1 Stiffness tests

We design a simple setup to test all possible stiffening criteria. A metric we introduce for this purpose is called Stitch Count Ratio (SCR). The SCR measures the number of stitches not aligned in the preferred direction to stitches aligned in the preferred directions (given by the normal map). SCR also loosely corresponds to the ω parameter we saw in Section 3 in mixing cost functions. When $\omega = 0.0$, we have complete anisotropy since all stitches follow the preferred direction from the normal map. The SCR value hence should also be close to 0.0, because the number of unaligned stitches is roughly 0. In the case of complete anisotropy, $\omega = 1.0$ and SCR is also roughly 1.0 because the number of unaligned and aligned stitches is equal.

We make use of a *reverse lookup table* to perform a simple line search to find the appropriate *alphas* and *betas* for our cost functions. We run the stitch planner with different values for *alpha, beta* and ω generated by performing an exhaustive search. Then, we handpick 3 different configurations of these parameters (the handpicked examples are also chosen based on visual appeal), one for when

Figure 4.1: Testing apparatus

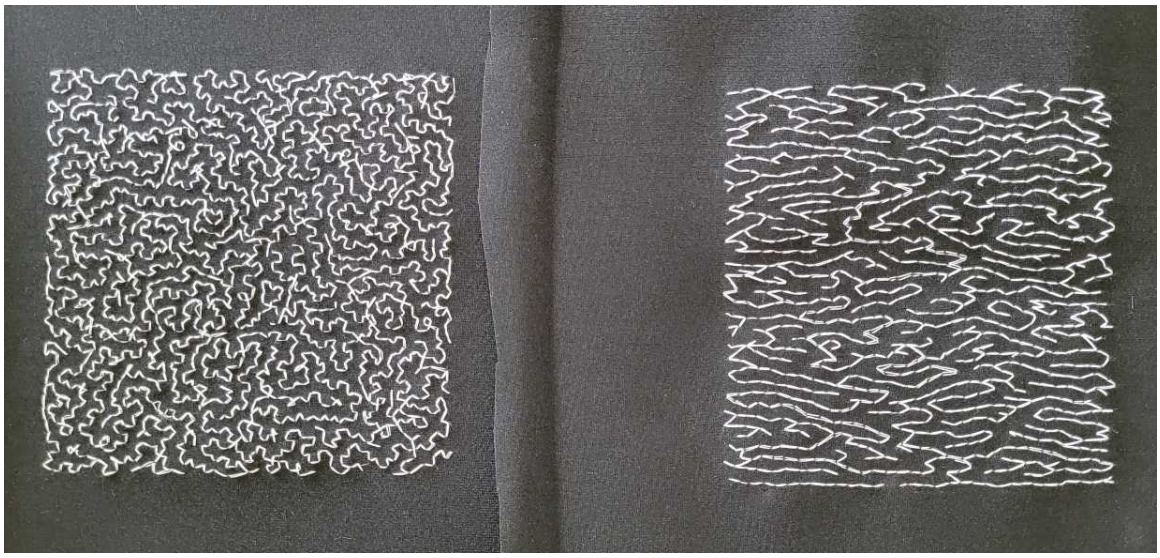

Figure 4.2: Left - fully isotropic planning, Right - fully anisotropic planning

SCR $\approx \omega \approx 0.0$, the other for SCR $\approx \omega \approx 0.5$ and finally for SCR $\approx \omega \approx 1.0$. The intermediate values for SCR can be computed by performing a simple linear interpolation between the parameter vectors, which also work well in practice as our experiments here will show.

Results of our first experiment can be seen in Figure 4.3. The stitch plans used for comparison can be seen in Figure 4.2. The ω values for the two plans were set to be 0.0 and 1.0 (complete anisotropy and complete isotropy). The results show that the completely anisotropic stitch plan resists stretching in the horizontal X direction, whereas stretches the most in the vertical Y. For the completely isotropic plan, the stretch in the X and the Y is roughly the same, giving us the results we expected.

In our second set of experiments, we blend ω values smoothly from 0.0 to 1.0. The target SCR our stitch plans produce also coincides with the value of ω. We impose a condition of a fixed number of horizontal stitches the stitch plans need to have, which we set to 1000, with the expectation of getting the same horizontal stiffening for all plans, but a variable stiffening in the vertical Y direction. To accomplish this, we augment the number of points sampled as we go from complete anisotropy to complete isotropy. The rationale for this is that for complete anisotropy, all stitches will be aligned with the preferred direction, giving us maximum stiffening in that direction. On the other hand, for complete isotropy, roughly half the stitches will be aligned with the horizontal

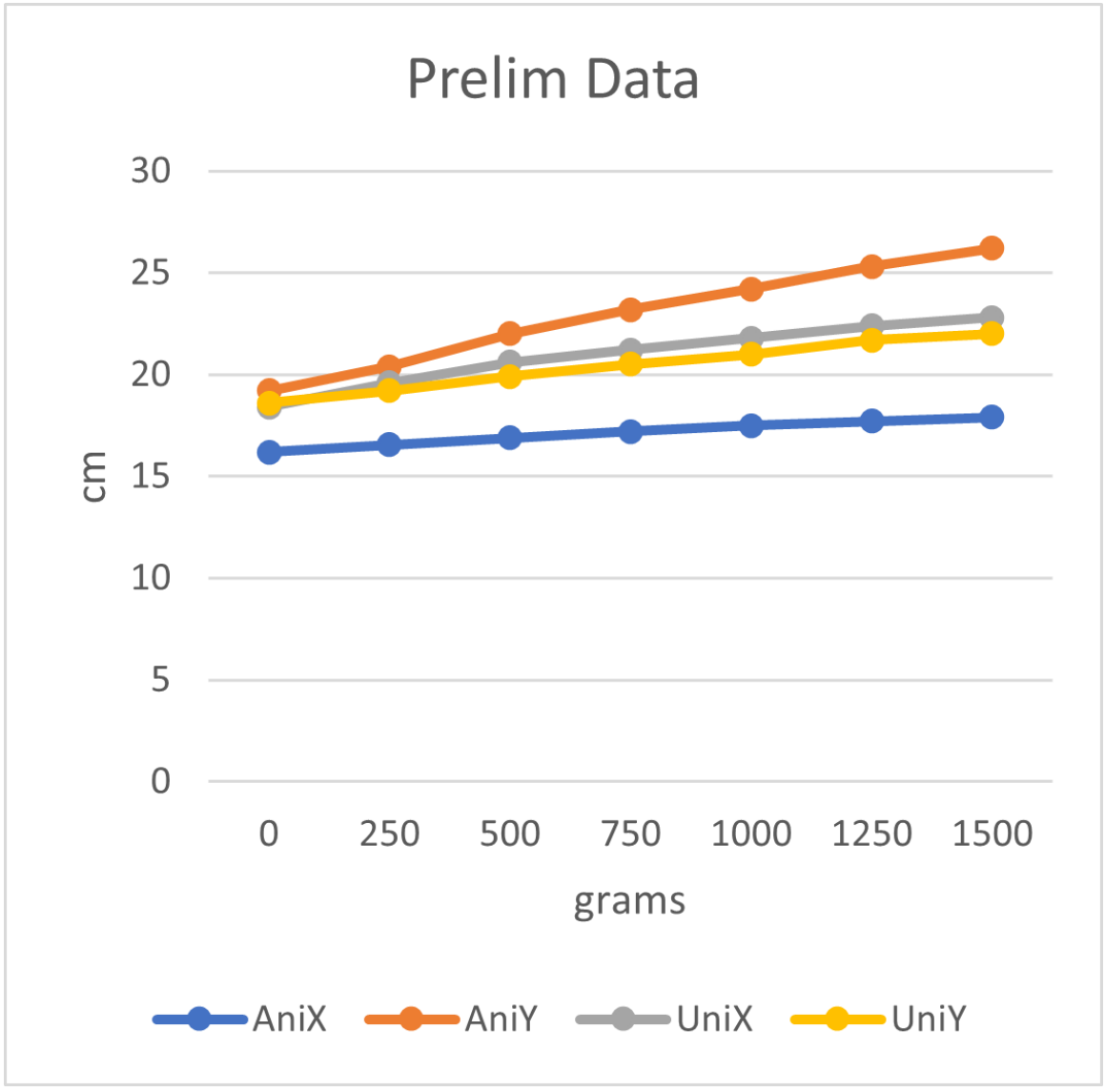

Figure 4.3: Stiffness comparison of the two extremes

direction and the other half would not, which implies a need to sample double the number of points in order to ensure the same horizontal stitch count of a 1000.

Let x be the number of points sampled for complete anisotropy. The formula for augmenting points is given by $x^{'} = (\omega + 1.0) \cdot x$. Figure 4.4 shows the difference in the stress-strain curves for different values of ω used. The solid lines denote stretching along the horizontal direction, which are roughly constant for all plans. The reason for them not being closer to each other is the noise introduced while performing the statistical analysis of the stitch counts. In our program, every stitch at a 45 degree angle or less with the preferred direction is classified as a horizontal stitch.

The test apparatus shown in Figure 4.1 consists of a variable set of weights attached to a hook connecting to the piece of fabric to be tested. The fabric used in these examples is 100 mm X 100 mm. A measuring scale on the left is used to gauge the displacement of the fabric induced by the weight.

4.2 Visual validation of anisotropic stitching

The reversed normal map described in Section 3 is a handy tool to visualize how well the stitch planner followed direction. An important thing to note is that this validation approach can only be used for qualitatively gauging fully anisotropic stitch plans when $\omega = 0.0$. Figures 4.7 and 4.8 demonstrate another visualization of the two approaches employed for stitch planning. We mark off-directional stitches with a different color (green and red).

4.3 Embroidery machine

We employ a Brother SB7900E professional embroidery machine to produce the embroidered stitch plans. Our experiments use a medium weight 4-way elastane fabric with 50 weight poly embroidery thread used to make the individual stitches. Based on the machine instructions, we tension both the material and thread before commencing the embroidery process. Using a water-soluble film backing to hold the fabric in place lead to reduction in knotting, as per our observations. The average time for embroidery of the different examples shown in this work runs between 15-30 mins depending on complexity and topology of the print.

The machine expects a .dst file as input, a well known embroidery file format. We employ

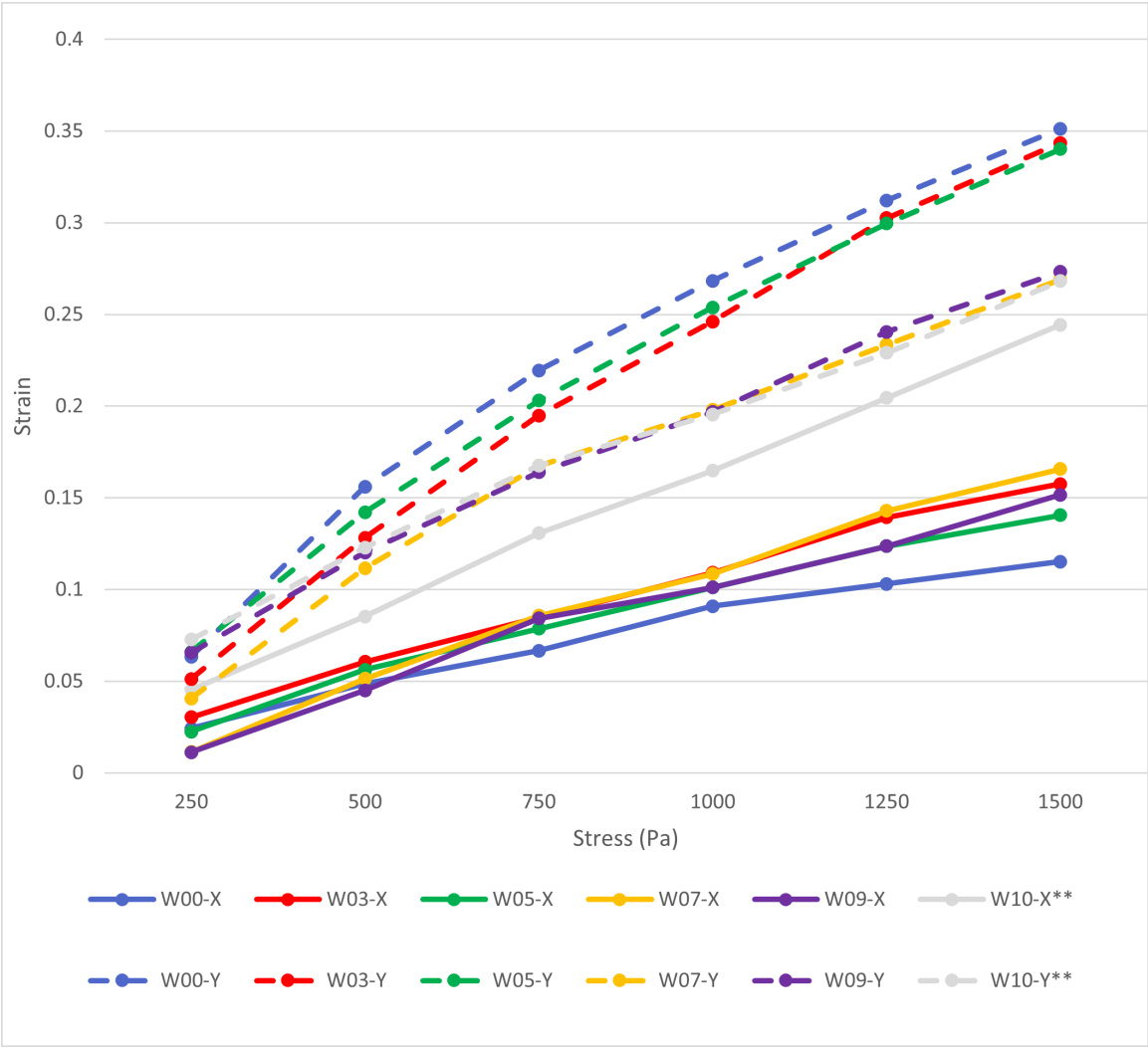

Figure 4.4: Stiffness comparison by blending omega

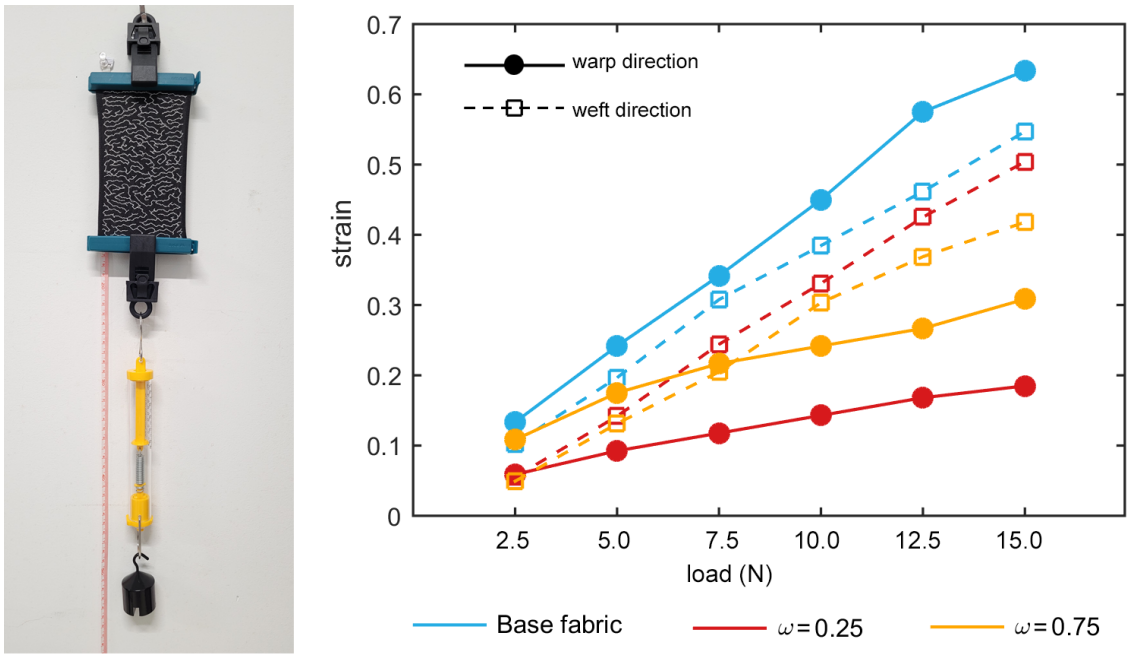

Figure 4.5: Stiffness comparison for omega = 0.25, and omega = 0.75

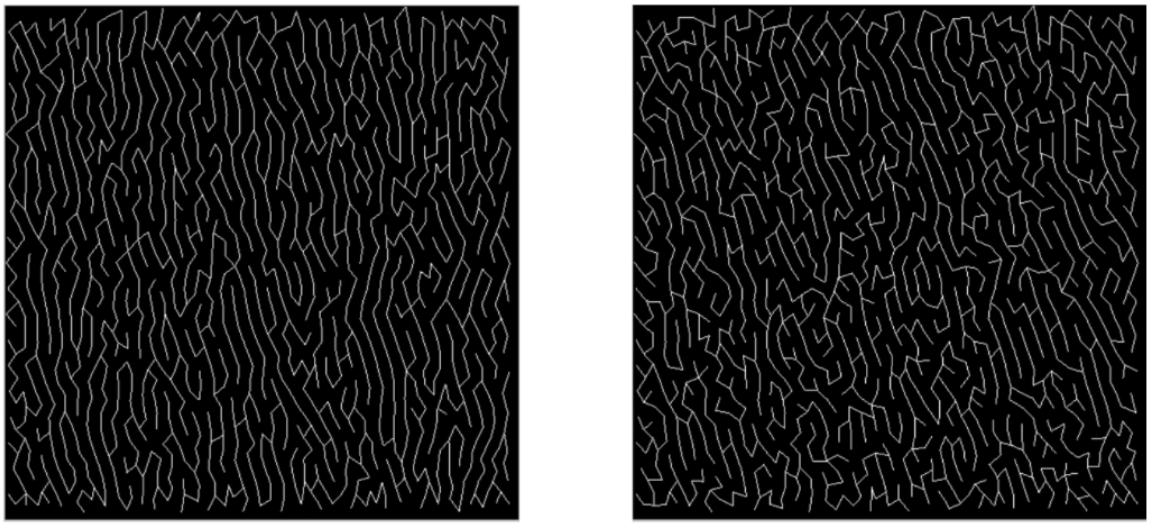

Figure 4.6: Stitch plans generated with omega = 0.25 and 0.75 respectively. The normal map enforces preferred direction to vertical for these examples. The sampling of density is uniform.

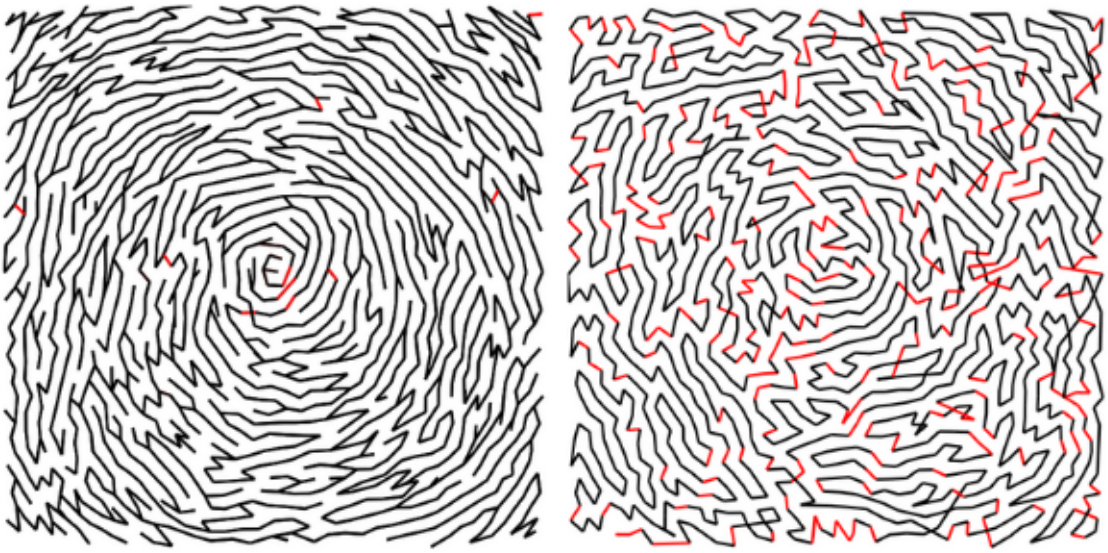

Figure 4.7: Left - Dijkstra's algorithm with the post processing cleanup routine, Right - TSP with anisotropic cost function. Red segments indicate off-directional stitches. Made with a uniform density sampling with the cone normal map.

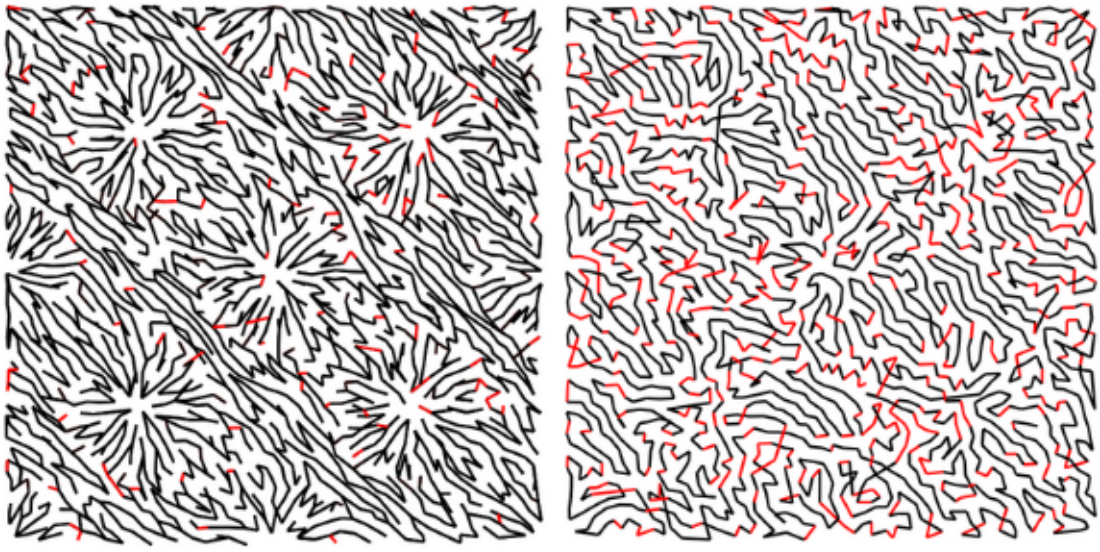

Figure 4.8: Left - Dijkstra's algorithm with the post processing cleanup routine, Right - TSP with anisotropic cost function. Green segments indicate off-directional stitches. Made with a uniform density sampling with the polka dot normal map.

Figure 4.9: The embroidery machine in action

a program from the EmbroiderModder project on github, to convert the CSV file produced by our stitch planning program to a dst file. The CSV file our program generates consists of a series of points interpreted by the machine as instructions on what points to visit for making stitches. Figure 4.5 shows the machine in action.

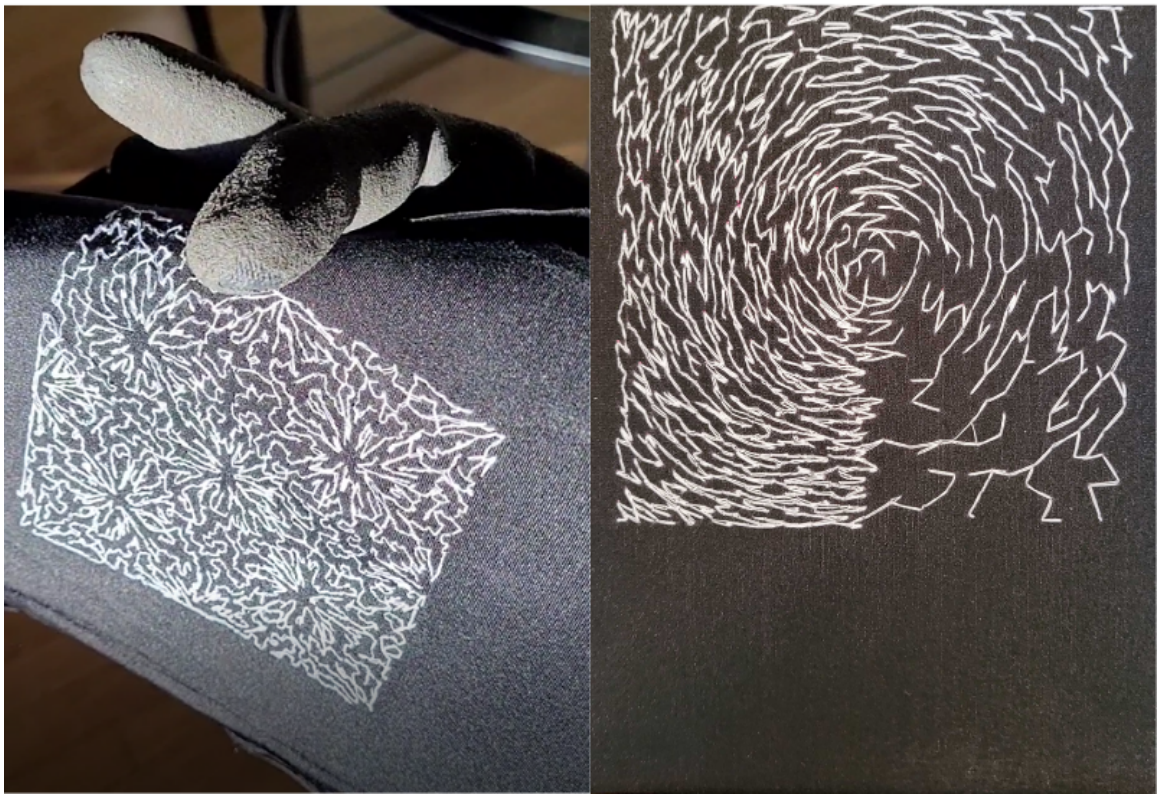

Figure 4.10: Left - Polka dot plan, Right - Cone plan

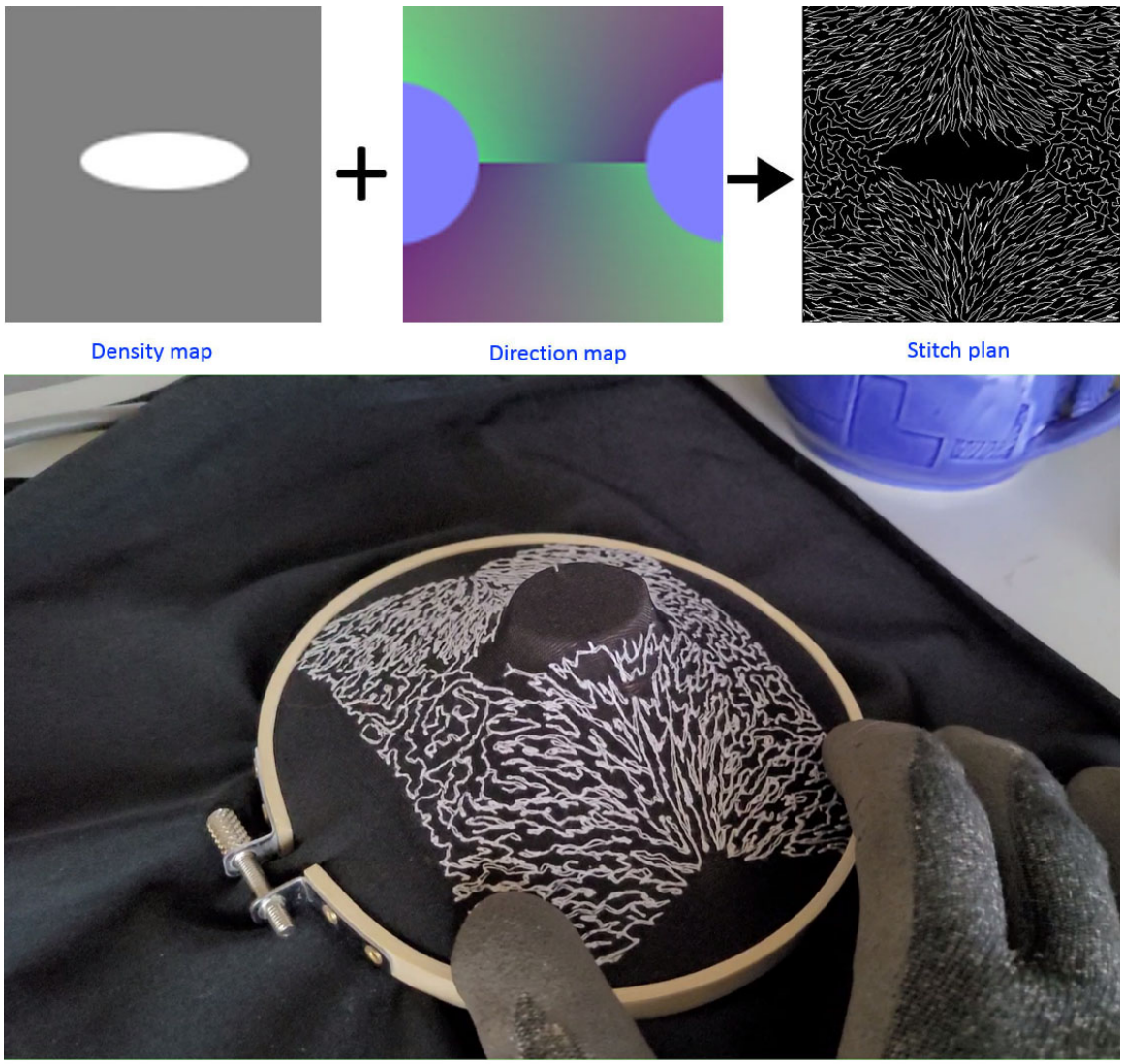

Figure 4.11: This example shows an anisotropic stitched fabric under a loaded condition. This image, and the video, show that the 3D shape of this tiny "tent" is affected by the stitching.

Chapter 5

Conclusions and Discussion

This work presented a framework for altering tensile properties of general purpose cloth fabric in a purposeful fashion. We demonstrated a novel path planning scheme for custom embroidery and an inverse design approach driven by optimization to assist users in their quest for finding the optimum designs for precisely specified performance criteria. We were also successful in generalizing the stitch planner from the previous work by [24] to produce a full range of material properties, all the way from fully isotropic to fully anisotropic. We believe these contributions will open up many exciting possibilities for applications in a variety of areas including healthcare, personalized smart clothing, construction, etc and look forward to making improvements and updates to our current work described in the following section. We will also look at the shortcomings of the present approach and potential solutions for enhancing them.

5.1 Future Work and Limitations

5.1.1 Stitch planning limitations

The stitch planning techniques described in this work have the ability to alter tensile properties of the underlying fabric material, but only with limited precision. The *Stitch Count Ratio* approach introduced to accomplish this task used a linear model to control the amount of anisotropy/isotropy. From our observations, it is clear that a series of consecutive stitches oriented in the same direction induce more stiffening than if they were disjointed. However, the *SCR* model

assumes same stiffness for both cases.

In our experiments, we also observed the formation of long straight rows of stitching at the boundaries of the sample and in low density regions. The boundary problem arises due to the CVT sampling technique, where points in some regions are pushed out of bounds (outside the 100X100 pixel square) to make them coincide with the centroids of the Voronoi regions. These points then have to be clamped so that they remain in bounds and end up forming unwanted collinear patterns. Straight long series of stitches are also more prone to snapping under bigger loads since a single stitch bears the entire load, although the double stitching made because of the tree does help alleviate this problem to some extent. If this artifact is found in a low density region, it induces unnecessary stiffening in that region, a property not desirable from a low density region.

5.1.2 Performance criteria specification

We specified performance criteria for the 2 examples shown in Section 3.2.5, by asking the optimizer to minimize distance between the top corners of the two rigid bodies (the table and cube) residing on the surface of the cloth. The optimizer would have given us a similar result, had we tried to minimize the angle between the top surface and the surface of the cloth (i.e parallel to the cloth surface). These criteria were hard-coded into the program we used for optimization. In the future, we would like to make it easier for the user to choose between different options and add more flexibility to the process. Features such as specifying a precise Young's modulus the cloth needs to have in the horizontal/vertical/shear direction would also be great additions.

5.1.3 Coherence between virtual and physical reality

One of the great challenges Computer Graphics researchers have faced ever since the inception of the field is to bridge the gap between the virtual and physical. While the requirement for most graphics applications is to be interactive and real time, accurate physics computations aren't always a must and can be faked using clever mathematical trickery. Computational fabrication applications pose another challenge altogether, since the simulations have to be very closely grounded in physics so they can be replicated with precision in the real world. The cloth simulator we used for our inverse design experiments was not calibrated with the physical cloth fabric we used for the stitch planning experiments and although we haven't yet fabricated physical examples yet, it is safe

to say that they will most likely not behave like their simulated counterparts. They do however produce intuitive results (Section 3) that at least visually seem correct (refer to the cube stitch plan in Section 3). An interesting future work would be to capture real cloth behaviour and replicating it in a simulator by fitting an elastic model, for predicting deformations on arbitrary forces applied to the surface of the cloth [23, 33]. Capturing the effect of embroidery on the cloth surface is another challenging and (probably the most important) task, which once solved, will truly help us bridge the gap between the virtual and physical. Measuring the corresponding Young's modulus and Poisson ratio's and storing them in a material browser to enable users to discover aesthetically pleasing stitch plans that produce desired mechanical behaviour similar to the work of [30] is another interesting path that can be taken moving forward.

Bibliography

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[2] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[3] James F Blinn. Simulation of wrinkled surfaces. *ACM SIGGRAPH computer graphics*, 12(3):286–292, 1978.

[4] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[5] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[6] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[7] Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM journal on numerical analysis*, 44(1):102–119, 2006.

[8] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[9] Rafael C Gonzalez, Richard E Woods, and Barry R Masters. Digital image processing third edition. *Pearson Prentice Hall*, pages 743–747, 2008.

[10] Ruslan Guseinov, Eder Miguel, and Bernd Bickel. Curveups: Shaping objects from flat plates with tension-actuated curvature. *ACM Trans. Graph.*, 36(4):64:1–64:12, July 2017.

[11] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[12] James C Hateley, Huayi Wei, and Long Chen. Fast methods for computing centroidal voronoi tessellations. *Journal of Scientific Computing*, 63(1):185–212, 2015.

[13] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017.

[14] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.

[15] Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. Knitting a 3d model. In *Computer Graphics Forum*, volume 27, pages 1737–1743. Wiley Online Library, 2008.

[16] Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. Knitty: 3d modeling of knitted animals with a production assistant interface. In *Eurographics (Short Papers)*, pages 17–20. Citeseer, 2008.

[17] Jenny Lin, Vidya Narayanan, and James McCann. Efficient transfer planning for flat knitting. In *Proceedings of the 2nd ACM Symposium on Computational Fabrication*, pages 1–7, 2018.

[18] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):1–17, 2009.

[19] Ali Mahdavi-Amiri, Philip Whittingham, and Faramarz Samavati. Cover-it: An interactive system for covering 3d prints. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, pages 73–80, Toronto, Ont., Canada, Canada, 2015. Canadian Information Processing Society.

[20] Jonàs Martínez, Jérémie Dumas, and Sylvain Lefebvre. Procedural voronoi foams for additive manufacturing. *ACM Trans. Graph.*, 35(4):44:1–44:12, July 2016.

[21] Jonàs Martínez, Haichuan Song, Jérémie Dumas, and Sylvain Lefebvre. Orthotropic k-nearest foams for additive manufacturing. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[22] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica Hodgins. A compiler for 3d machine knitting. *ACM Trans. Graph.*, 35(4):49:1–49:11, July 2016.

[23] Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. Data-driven estimation of cloth simulation models. In *Computer Graphics Forum*, volume 31, pages 519–528. Wiley Online Library, 2012.

[24] Ella Moore, Michael Porter, Ioannis Karamouzas, and Victor Zordan. Precision control of tensile properties in fabric for computational fabrication. In *Proceedings of the 2nd ACM Symposium on Computational Fabrication*, pages 1–7. ACM, 2018.

[25] Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James McCann. *ACM Trans. Graph.*, 37(4), July 2018.

[26] Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. Visual knitting machine programming. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[27] Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. Elastic textures for additive fabrication. *ACM Trans. Graph.*, 34(4):135:1–135:12, July 2015.

[28] Huaishu Peng, Scott Hudson, Jennifer Mankoff, and James McCann. Soft printing with fabric. *XRDS: Crossroads, The ACM Magazine for Students*, 22(3):50–53, 2016.

[29] Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. Design and fabrication of flexible rod meshes. *ACM Trans. Graph.*, 34(4):138:1–138:12, July 2015.

[30] Christian Schumacher, Steve Marschner, Markus Gross, and Bernhard Thomaszewski. Mechanical characterization of structured sheet materials. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

[31] Georgi Stoychev, Mir Jalil Razavi, Xianqiao Wang, and Leonid Ionov. 4d origami by smart embroidery. *Macromolecular rapid communications*, 38(18), 2017.

[32] Spencer W Thomas and Rod G Bogart. Color dithering. In *Graphics Gems II*, pages 72–77. Elsevier, 1991.

[33] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)*, 30(4):1–12, 2011.

[34] Rundong Wu, Claire Harvey, Joy Xiaoji Zhang, Sean Kroszner, Brooks Hagan, and Steve Marschner. Automatic structure synthesis for 3d woven relief. *ACM Transactions on Graphics (TOG)*, 39(4):102–1, 2020.

[35] Jonas Zehnder, Espen Knoop, Moritz Bächer, and Bernhard Thomaszewski. Metasilicone: design and fabrication of composite silicone with desired mechanical properties. *ACM Transactions on Graphics (TOG)*, 36(6):240, 2017.

[36] Xiaoting Zhang, Guoxin Fang, Mélina Skouras, Gwenda Gieseler, Charlie Wang, and Emily Whiting. Computational design of fabric formwork. 2019.